修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院 情報理工学工学研究科 情報・ネットワーク工学専攻 博士前期課程		
氏　　名	寺田　充樹	学籍番号	1831103
論 文 題 目	養蜂のためのカメラ画像を用いた外敵の検出に関する研究		

要　　旨

本研究ではIoT技術をこれまで経験とカンに頼ってきた養蜂に導入し，プロでも現地で巣箱の内外を観察しなければわからないことを遠隔でモニタするシステムの開発と評価を行った．外気および巣箱内の温湿度，重量，二酸化炭素濃度を計測し，スマートフォン等でその変化をモニタするセンサシステム，そして早期対応が必須であるスズメバチの襲来を機械学習モデルによって検出するカメラシステムを，安価なマイコンボード ESP32 およびRaspberry Piを用いて実装した．温湿度センサ，重量センサ，二酸化炭素センサは試験的に1群のみにセンサを設置したが，そこではダニや分蜂等の異常は発生しなかったため異常検知はできなかった．しかし，遠隔モニタで巣箱の温湿度や二酸化炭素濃度は一定に保たれ正常であることや重量変化からミツバチの群勢や採蜜時期の判断材料になるなどの有用性が示された．機械学習を用いたスズメバチの物体認識では Web の写真やビデオ撮影した動画から切り出した 500 枚の画像データを用いて，ImageAI，TensorFlow，TensorFlow Lite環境で比較評価した．TensorFlow Liteは42.6%の精度で他よりも低いが処理速度が ImageAI の 20 倍，TensorFlow の 30 倍以上高速であった．そこで，学習データを900枚増やしたデータセットでは92.9%の検出率を得ることができた．誤検出も多少生じたが，検出を正解とする閾値を高くすることで誤検出をなくすことができることを示した．また，Raspberry Piの各モデルやエッジAIデバイスで処理速度・容量を比較し，価格面や複数台のカメラを接続させた時のFPSを考慮すると7,700円で10台のカメラを1台あたり1fps以上で処理できるRaspberry Pi 4 model Bが適当だった．実際に養蜂場に設置した想定で防水ウォールボックス内にRaspberry Pi 4 model Bとモバイルルータを入れ，ESP32 と距離を離した時の 8m 以上距離を置くと接続が切れることがわかった.さらに,スズメバチの検出の手法をミツバチの個体数のカウントに応用し,誤差7~22%という結果が得られた．精度向上に向けてまだ多くの改善の余地が残されているが，これは先行研究の誤差2倍以上と比較して極めて高い精度である．また，この結果からIoTやAIを応用した養蜂の研究が，いかに初期の段階であるかを物語っている．機械学習による映像解析は，巣門前の個体数だけでなく，採蜜のために出入りするミツバチの数だけをカウントすることで活動状態を調べたり，巣箱を開けて巣枠に密集している蜂の状態を全体で観察して群の個体数を把握したり，またそれぞれの働きバチが持つ役割を解析したりと様々な応用が考えられる．本研究ではその大きな可能性を示すことができた．

令和元年度修士論文

養蜂のためのカメラ画像を用いた外敵の検出に関する研究

電気通信大学情報理工学研究科
情報・ネットワーク工学専攻
コンピュータサイエンスプログラム

学籍番号：1831103
氏名　　：寺田 充樹
指導教員：佐藤 証 教授
提出日　：2020 年 3 月 18 日

要旨

ミツバチは世界の食料の 9 割を占める作物種の受粉に携わり，その 7 割を担っていると言われる．養蜂は持続可能な社会の実現を目的として国や企業だけでなく一般にも浸透してきている SDGs の活動に大きく貢献し，また個人でも行える唯一の畜産業として静かなブームとなっている．また近年，都市農業が注目されるとともに，都市養蜂も盛んとなってきている．

農業の IoT 化は大規模農場や植物工場には浸透してきている一方，養蜂の IoT 化に関する研究は行われているが製品は世界でもほとんど見当たらず，国内では温湿度センサが一社から販売されているだけである．ミツバチをカメラで観察することにより，センサの数字だけではわからない情報が得られ，またスズメバチの襲来の検知にも映像は不可欠であるが，カメラを実装した養蜂用の製品はない．

そこで，本研究ではこれまで経験とカンに頼ってきた養蜂に IoT 技術を導入し，プロでも巣箱の内外を観察しなければわからないことを，遠隔でモニタするシステムの開発と評価を行った．外気および巣箱内の温湿度，重量，二酸化炭素濃度の変化を，スマートフォン等で監視するセンサシステム，早期対応が必須であるスズメバチの襲来を機械学習モデルで検出するカメラシステム，さらにミツバチの活動状態を把握するための個体数計測システムを，安価な IoT 用マイコンボード ESP32 および Raspberry Pi を用いて実装した．これらを養蜂箱に設置して屋外での安定動作の検証を行うとともに，養蜂施設で撮影したスズメバチやミツバチの動画による検出精度評価を行った．

養蜂施設は草原や山間部を利用したり，また季節の花を求めて各地を転々とする移動養蜂もある．そのため固定のブロードバンドネットワークの利用は困難で，IoT の導入ではモバイルルータによる携帯電話回線の利用が前提となる．その場合，動画等の大きなデータをクラウドサーバに送信することは難しく，ローカルでデータを処理するいわゆるエッジコンピューティングの実現が重要である．そこで Raspberry Pi のローカルサーバに機械学習ライブラリ TensorFlow Lite を実装し，複数カメラを接続して実用的な速度や安定したネットワークが組めるか等についても評価を行った．

温湿度センサを設置した群ではダニや病気等の異常や分蜂は発生しなかったため，温湿度データから異常の判断ができるかどうかは不明であったが，常時安定した数値であることは確認できた．また重量センサからは，春の採蜜期に巣箱の重量が日々増加していく様子がはっきりとわかり，一群あたり 5~6 回の収穫時期を判断するのに有用であるとの評価が利用者から得られた．

機械学習によるスズメバチの検出では，複数の学習データから GPGPU パソコンで生成したモデルを ImageAI，TensorFlow，TensorFlow Lite の実行環境で比較し，いずれも良好な結果が得られた．エッジコンピューティングの実現において重要となるモバイルデバイス環境の TensorFlow Lite は，精度は他より低いものの処理速度は ImageAI の 20

倍, TensorFlow の 30 倍以上となった. 学習データを増やすことで検出精度の向上を図ったところ，検出率が 42.6%から 92.9%に増加した．そして，誤検知についても閾値を定めることで実用においてゼロにできる可能性を示した．また，TensorFlow Lite を Raspberry Pi 4 model B / 3 model B / Zero W の 3 機種に実装し，Coral USB Accelerator も用いて 10 台の ESP32 カメラを接続したときの性能および価格について検討も行った．その結果，Raspberry Pi 4 model B と Coral USB Accelerator の性能がほぼ互角となり，前者が対価格性能比で最も優れていることを明らかにした.

さらに，スズメバチの検出手法をミツバチの個体数カウントに応用したところ，誤差 7~22%という結果が得られた．精度向上に向けてまだ多くの改善の余地が残されているが，誤差 2 倍以上という先行研究と比較して極めて高い精度である．この結果は IoT や AI を応用した養蜂の研究が，いかに初期の段階であるかを物語っている．機械学習によるミツバチの映像解析は, 巣枠に密集している蜂の状態を全体で観察して群の個体数を把握したり，採蜜のために出入りするミツバチだけをカウントし，働きバチ毎に決められている役目を検出・分類することで活動状態を調べるなど様々な応用が考えられる．本研究はその大きな可能性を示すことができた.

目次

1　背景と研究目的 1
1.1　背景 1
1.2　研究目的 2
1.3　論文の構成 2

2　都市養蜂 4

3　従来技術と関連研究 7
3.1　養蜂の IoT 製品 7
3.2　先行研究 9

4　使用技術 13
4.1　マイコンボード 13
4.2　センサデバイス 15
4.3　エッジ AI デバイス 18
4.4　機械学習による物体検出 18
4.5　物体検出ライブラリ 20

5　センサシステムの開発 22
5.1　概要 22
5.2　重量・温湿度センサシステム 22
5.3　二酸化炭素濃度計測 26
5.4　巣箱内カメラシステム 27

6　スズメバチ検出システム 29
6.1　概要 29
6.2　スズメバチの動画撮影 29
6.3　学習用データセットの作成 31
6.4　機械学習によるスズメバチ検出 34
6.5　Raspberry Pi での速度評価 38
6.6　検出の精度向上 39
6.7　複数カメラによるスズメバチ検出 41
6.8　モバイルルータによる通信評価 44

7　ミツバチの個体数計測 46
7.1　概要 46
7.2　学習用データセットの生成 46
7.3　機械学習によるミツバチの個体数計測 47

8　むすび 53

参考文献 54

謝辞 58

1 背景と研究目的

1.1 背景

近年, IoT や AI 技術の農業や畜産業等の第 1 次産業への活用が広がりを見せている．そこでは高齢化と後継者不足を解決するために熟練の知見をデータ化し, また作業の効率化を図るために, 各種センサにより現場状況をリアルタイムでモニタリングして意思決定支援や作業の効率化が試みられている[1][2][3]. 農業分野では植物工場への導入や, 様々なセンサや監視カメラが製品化されている．しかし，センサ単体でも数万円，システムとなると数十万~数百万円と高価なため大規模農場でしか利用することができない. また，大規模農場は地価の安い地方にあるため，通信インフラが整っていないことも多い．特にカメラ映像のデータは大きいため施設に複数台設置する場合，専用回線を引くか各カメラに携帯電話網の LTE モジュールを接続する等，通信コストも導入への大きな障壁となる．

これまでの IoT 技術は，センサからの情報をクラウドサーバに集約して，ビッグデータ解析を行うという方向で研究が進んでいた．しかしながら，IoT デバイスの導入をありとあらゆるアプリケーションに広げようとする中で, 通信の速度や安定性そしてクラウドサーバの処理能力等が問題となってきている．そこで，データの収集時に前処理を行って通信量を減らす, あるいはその現場にサーバを配置することでリアルタイム処理を行うエッジコンピューティングが注目されている. つまりクラウドコンピューティングは，サーバにデータを集約する集中処理型であるのに対して，エッジコンピューティングは，ローカルな計算資源を活用する分散処理型となる．ローカルサーバを置くことはコスト的に不利であり，専門家でないと設定が難しいという懸念もあるが，マイコンの高性能化と低価格が進み，誰でも気軽に利用できる環境が整いつつある．例えば，教育用途から始まり実用でも広く利用されている Raspberry Pi[4]は，数千円と安価ながら 64 ビットの 4 コアモデルも登場し，一昔前のパソコンにも匹敵する処理能力を有している. また, IoT デバイスとして広く使われ互換機も多数開発されている Arduino[5]は，8 ビットマイコンからスタートしたが，現在は Wi-Fi と Bluetooth を搭載した 32 ビットマイコン ESP-WROOM-32 を搭載したものが 500 円程度で入手できるようになっている．このような高性能・低価格マイコンを用いたエッジコンピューティングにより，これまで IoT 化が進んでいなかった中小規模の農地での利用や, 新たなアプリケーションへの活用が期待される．

1.2 研究目的

佐藤研究室ではこれまで，無線センサを搭載した水耕栽培システムを開発し，ビルの屋上やベランダ等の空きスペースを活用した都市農業の研究を進めてきた. 都市農業は, 国連が 2015 年に採択した『持続可能な開発目標(SDGs: Sustainable Development Goals)』[6] の実現に向けて国や企業が取り組む中で，環境・社会・経済のバランスを重視した持続可能な都市・地域づくりの一環として重要視されている．また，都市農業に限らないが，世界の食料の 9 割を占める作物種の 7 割でハチが受粉を媒介しており，養蜂なくして農業は成り立たない．養蜂は SDGs の実現に大きく貢献し，個人でも行える唯一の畜産業として静かなブームとなっている．しかしながら，IoT 化はまったく進んでおらず，養蜂家のカンのみに頼っていると言っても過言ではない．またプロであっても養蜂箱内部の状態は蓋を開けてみなければ把握することができない. これはミツバチにとって良いことではなく, 特にまだ群勢が弱く気温の低い春先には大きなストレスを与える. 巣箱の外からの観察もミツバチの状態を把握したりスズメバチ等の外敵への対応に重要であるが，一日中巣箱を観察することは極めて困難である.

そこで本研究では，カメラおよびセンサデータを ESP-WROOM-32 で取得し，Raspberry Pi によってローカル処理する養蜂のためのエッジコンピューティングシステムを構築する．温湿度センサ，重量センサ，二酸化炭素センサ，赤外線カメラ等を養蜂施設に設置し，巣箱の内部と外部を 24 時間監視し，特に外敵の検知に関しては AI による機械学習を導入する．そして安価なデバイスで，どこまでの性能が得られるかを検証する．またネットワークや電源が安定した屋内での実験とは異なり，風雨にさらされ通信状態も悪い屋外で，安定した運用が可能かどうかの検討も行う.

1.3 論文の構成

次章以下，論文の構成は以下のとおりである.

第 2 章ではまず，本研究の実験を行っている都市養蜂と，養蜂に用いられるセイヨウミツバチとニホンミツバチ，そして最大の外敵であるスズメバチについて説明する.

第 3 章では，センサ等の IoT 技術を利用した養蜂製品とミツバチの個体検出に関する先行研究を紹介する.

第 4 章では，本研究で利用したハードウェアと，機械学習による物体検出について記す.

第 5 章では，それらを利用して構築したセンサシステムと，都内の養蜂施設で行った実験について述べる.

第 6 章では，各種機械学習モデルによるスズメバチ検出の精度の評価を行い，特にモバイルデバイスに特化した TensorFlow Lite による精度を向上させ，複数台のカメラによる同時検出の能力について検証実験を行う.

第 7 章では，スズメバチの検出手法をミツバチの個体数カウントに応用することで，先行研究に対する大きな優位性を示すとともに，機械学習によるハチの映像解析研究の大きな可能性について述べる．

最後に第 8 章で，本研究をまとめるとともに，今後の展望について述べる．

2　都市養蜂

近年，都市部のビルの屋上を利用して養蜂を行う「ミツバチプロジェクト」が盛んとなっている．国内では 2006 年に東京・銀座で始まったこのプロジェクト[7]は，池袋[8]，赤坂[9]，原宿[10]などに広がり，都内だけでなく，札幌[11]，仙台[12]，名古屋[13]など全国的な広がりを見せている．その運営も，NPO 法人，商工会，学校法人，企業など多種多様で，地産地消の蜂蜜を用いた地域活性化や環境教育などの取り組みが行われている．日本全国に広がっているミツバチプロジェクトは，蜂蜜の採取だけが目的ではなく，採蜜体験や講習会開催の「環境教育」，採取した蜂蜜を使った「地域ブランド開発」，緑化の啓発や緑化活動の「緑化促進」，そして周辺の蜜源マップ作成の「蜜源調査」のなど様々である．

図 2.1 は本研究に協力いただいた，TBS 屋上の養蜂施設「赤坂みつばちあ」[9]　と，原宿竹下通り入口の商業施設「はらじゅくアッシュ」屋上の「BEETOPIA はらじゅく」[10]である．

図 2.1　「赤坂みつばちあ」と「BARETOPIA はらじゅく」の養蜂施設

また，趣味や副業としての養蜂を楽しむ「週末養蜂家」も急増している．表 2.1 は国内の養蜂の状況で[14]，平成 26 年に数が急増しているが，これは平成 25 年から養蜂振興法により届け出が義務化されたためである．実際には届け出ていない個人養蜂家が数倍いるとも言われている．

表 2.1　蜜蜂飼育戸数，蜂群数　（単位：戸，千群，群／戸）

区分	H25 年	H26 年	H27 年	H28 年	H29 年	H30 年	R 元年
飼育戸数	8,312	9,306	9,567	9,452	9,395	9,578	9,782
蜂群数	204	210	213	212	213	213	215
平均蜂群数	24.5	22.5	22.3	22.4	22.8	22.2	22.0

世界の主要都市においても養蜂は盛んで[13]，フランスのパリ市内ではオペラ座，エッフェル塔，ホテルリッツなど 8 箇所で養蜂を行い，パリ産の蜂蜜として人気を集め

ている．米国のサンフランシスコ市内では養蜂登録者が最近 100 を超え，ニューヨーク市のマンハッタンには 400 を超える巣箱があると推定され，ホワイトハウスでも養蜂が営まれている．

国内の養蜂では，大きく分けてセイヨウミツバチ(Apis mellifera)とトウヨウミツバチ(Apiscerana)の一亜種であるニホンミツバチ(Apis cerana japonica Rad)の 2 種が利用されている．表 1.2 に 2 種類のミツバチの比較を示す．両者の飼育方法は生体の違いから異なる．セイヨウミツバチの養蜂箱は採蜜時に遠心分離機を使用できる人工巣枠が用いられ，採蜜後も巣脾を再利用する近代的な巣枠式巣箱が主に用いられている．ニホンミツバチの養蜂箱は採蜜時に巣脾を全て壊してしまう重箱式巣箱が主に用いられている．そのため，1 年に何度も採蜜を行えるセイヨウミツバチの採蜜量はニホンミツバチの 5 倍以上であり，養蜂業では主にセイヨウミツバチを飼育している．しかし，セイヨウミツバチは外来種を改良した家畜であり，日本の気候や病気に弱く，ダニ等の害虫やスズメバチ等の害敵への耐性がない．

表 2.2　セイヨウミツバチとニホンミツバチの比較

セイヨウミツバチ	ニホンミツバチ
	
① 採蜜量：多 ② 行動範囲：半径 4km ③ ダニや病気，害敵への抵抗力が弱い ④ 巣枠型巣箱	① 採蜜量：少 ② 行動範囲：半径 2km ③ ダニや病気，害敵への抵抗力がある ④ 重箱式巣箱

セイヨウミツバチは週に 1 回以上の巣内点検を行って，群の健康状態を把握し害虫の発生はいち早く対処する等，高い飼育技術が求められる．一方でニホンミツバチは野生の状態に近い形式で飼育するため，高い飼育技術や小まめな世話は必要がない．しかし，外敵の襲来が増えたり環境が悪くなるとすぐに群ごと移動する生態があるため，やはり頻繁に巣箱を観察する必要がある．

スズメバチは昆虫界のギャングとも言うべき凶暴性を有し，ミツバチを襲撃するのは主に図 2.2 のオオスズメバチとキイロスズメバチである．キイロスズメバチは単独で行動し，ミツバチを一匹咥えただけで飛び去るため被害はさほど大きくない．それに対し

てオオスズメバチは十数~数十匹で襲撃し，次々とミツバチを噛み殺すため致命的な打撃を与える．オオスズメバチの攻撃は以下の三段階であることが知られている[15].

第一段階でスズメバチは，巣箱を発見すると一匹ずつミツバチを捕まえて噛み砕き，肉団子にして巣へ運ぶことを繰り返す．トウヨウミツバチはスズメバチよりも生存できる温度が高いので，スズメバチに集団で群がる蜂球を作って熱殺する対抗手段を持っている．それに対してセイヨウミツバチはそのような防御法を持たず，単独で戦いを挑むがまったく歯が立たない．

第二段階は，十匹~数十匹で特定の巣箱に攻撃を集中する．ミツバチは大量に殺戮され，オオスズメバチの巣内への侵入を許すに至る．ここまで放っておくと手遅れである．

第三段階では，巣内に侵入したオオスズメバチは，サナギを引っ張り出して巣へ持ち帰る．オオスズメバチは普段は人に攻撃を与えないが，この段階では巣箱の所有者意識を持っており，集団で防衛しようとするため近づくのも危険となる．

都市部は地方ほど多くないがスズメバチは生息しており，もし襲撃されるようなことがあればその第一段階を察知して，駆除することが極めて重要となる．

図 2.2　オオスズメバチ(左)とキイロスズメバチ (右)

3　従来技術と関連研究

3.1 養蜂の IoT 製品

養蜂向けの IoT 製品は，筆者の知る限り国内外で次の 3 つが販売されている．

- BuzzBox [16]

OSBeehives 社が開発した温度(0~65℃±0.2℃)，湿度(0~100%±2%)，音(2Hz~3,150Hz)を測定するデバイスである．図 3.1 左の Buzbox は巣箱の外面や巣門の上に設置し，図 3.1 中央の小型番 BuzzBox mini は巣枠に取り付け，巣内の測定を行う．取得したデータは Wi-Fi でクラウドサーバに送信し, 専用のスマートフォンアプリでモニタできる. また, BuzzBox の設置場所を登録するとその地域の天気とリンクされる. 電源はソーラーパネル(3.75W)からバッテリー(500mh)に供給され，GPS による盗難防止システムも組み込まれている．価格は BuzzBox mini が$199 で，アプリは無料である．ミツバチの状態を把握する AI を導入しているとのことであるが，その詳細は不明である．

図 3.1　BuzzBox(左)，BuzzBox mini (中央)，専用アプリ(右)

- HIVE-TECH [17]

3Bee 社の図 3.2 のデバイスで，巣箱の下に重量センサを置き，巣門から音センサ，温湿度センサ，VOC(揮発性有機化合物)センサが入ったセンサ箱を挿入する．測定した

図 3.2　HIVE-TECH(左)と専用アプリ(右)

データは 1 日に 4 回インターネットを経由してクラウドサーバに送信され，スマートフォン等で専用のアプリからモニタできる．価格は 365 ユーロで，オプションとして GPS から巣箱の追跡システムもある．データはアプリでグラフ化され，生産量や新女王の誕生，病気の発生，分蜂などの推定を行うことができると説明がある．スタートアップであり，まだプレオーダーの段階のようである．

- Bee Sensing [18]

アドダイス株式会社が開発した国産製品で，温湿度センサで取得した情報とスマートフォンで入力した作業履歴情報をクラウドの AI で学習させ，蜜蜂の健康状態を診断するシステムである．図 3.3 の親機と子機で構築され，親機は AC100V アダプタ，巣箱に設置する子機は電池で駆動される．子機と親機の最大通信距離は 12m である．分蜂やダニを人為的に発生させた時の温湿度の波形をディープラーニングで学習させる実験で，分蜂は 83.4%，ダニの発生は 59.2%の精度で検出ができたと報告されている[19].

図 3.3　Bee Sensing の親機(左)と子機(右)

上記の製品は温湿度等のセンサデータを AI で学習させ，異常を検出するというものであるが，Bee Sensing 以外ではその詳細や具体的な数字が示されていない．また，分蜂やダニの発生等の巣内状態の見極めは，プロの養蜂家でも難しく，まだ研究の余地が多く残されている．特に動画による観察は，直接的にミツバチの活動状況が把握でき，またスズメバチ等の外敵の襲来もわかるが，カメラを備えた製品はない．また国内の養蜂の研究は玉川大学以外ではほとんどなく，玉川大学は生態に関する研究が中心で，IoT システムの導入などは行われていない．

3.2 先行研究

以下では，映像を用いた養蜂の研究について紹介する．

- 画像処理によるミツバチの計数 [20]

Kulyukin らは，巣門前のミツバチの個体数をカウントするシステムを開発している．Raspberry Pi 3 に接続したカメラで巣箱の上から巣門の撮影を行い，その画像内のミツバチを以下の手順で検出する．

1. 巣門入り口にある着地台周辺の画像を切り取る(図 3.4)

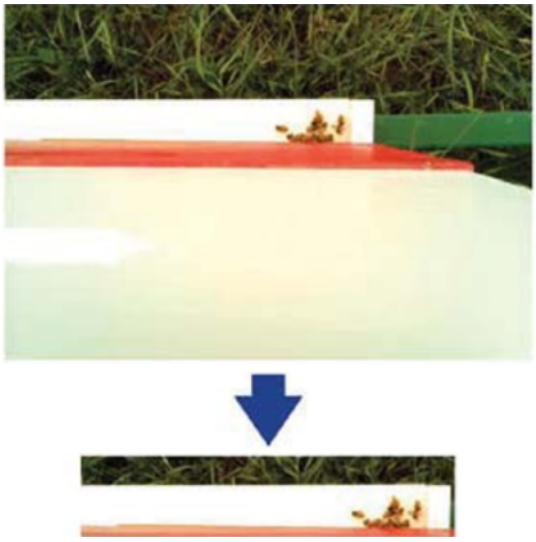


図 3.4　着地台付近の切り取り[20]

2. 輝度を 45~95 以内に調整
3. RGB 画像を HSV 画像に変換(図 3.5①)
4. 着地台の色である緑色を白領域にするため，opencv ライブラリの inRange()を用いて 2 値化 (図 3.5②)
5. ノイズを除去(図 3.5③)

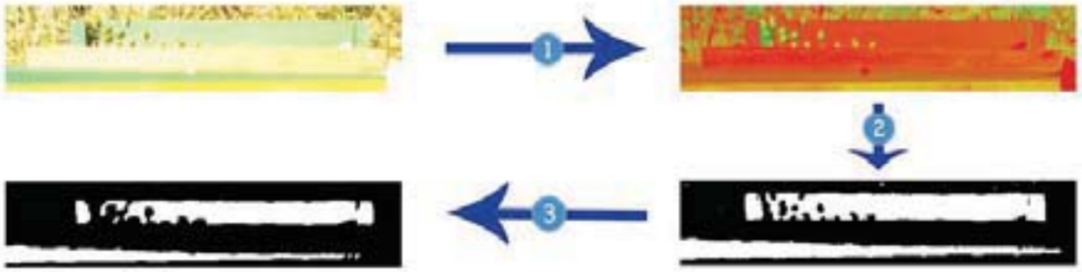


図 3.5　2 値化とノイズ除去[20]

6. opencv の findContours()を用いて白領域を長方形で囲む(図 3.6①)
7. 囲まれた領域から着地台の面積に近いものをピクセル数から絞り出す(図 3.6②)

8. 選択された領域でトリミング(図 3.6③)

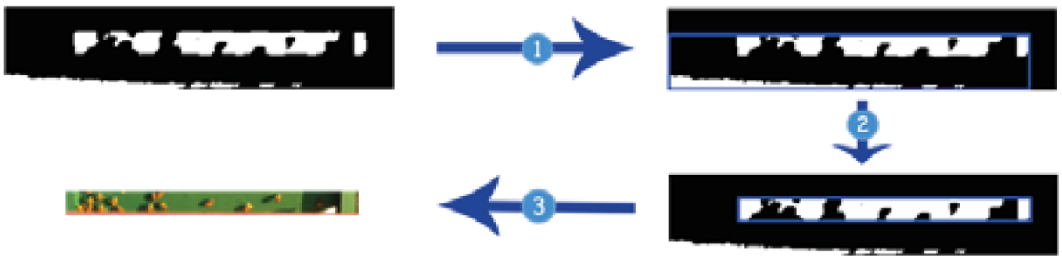


図 3.6 着地台をトリミング[20]

9. 着地台の画像から背景(白)と前景(黒)で 2 値化(図 3.7)

図 3.7 背景(白)と前景(黒)で 2 値化[20]

10. 黒領域をミツバチ 1 体のピクセル数で除算し，ミツバチの個体数を求める．

精度は着地台が緑色の場合 80.5%，白色の場合 85.5%という結果だった．このミツバチの個体数カウントシステムではカメラと着地台の距離や角度，着地台の大きさによって閾値を計算する必要がある．また，巣門前にミツバチが集中していたり，巣箱の側面にいるミツバチをカウントすることはきわめて困難である．

- 動体検出によるミツバチの計数 [21]

Kulyukin らは，動体検出と画像分類を用いて巣箱を出入りするミツバチを検出する図 3.8 の手法を提案している．動体が検知された領域を正方形で切り取り，ミツバチの有無を画像分類機に判定させて個体数をカウントする．動体検出には OpenCV の背景差分アルゴリズム KNN，MOG，MOG2 を試し，影の影響が少ない MOG2 の性能が高いことを示した．

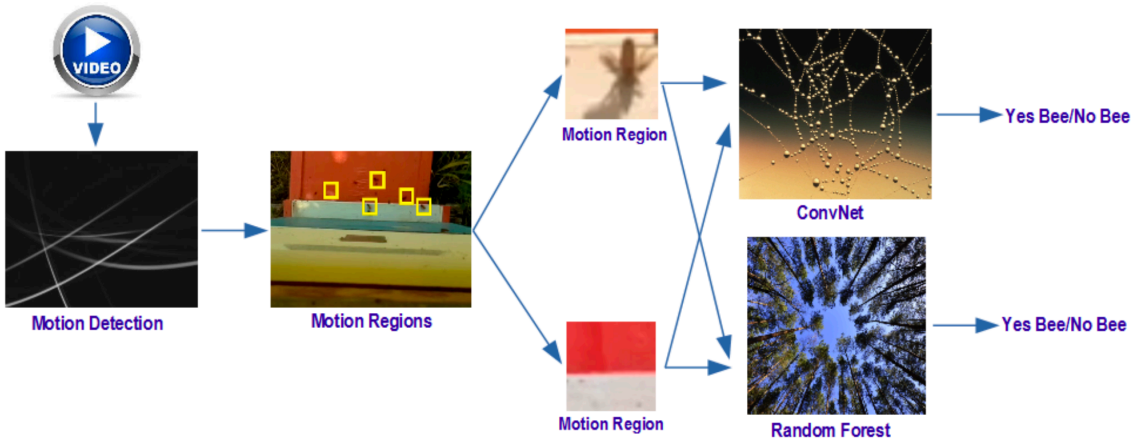


図 3.8 Kulyukin らの手法[21]

画像分類には，表 3.1 の 3 つのデータセットを用意している．BEE1 は 32×32 画像を 54,382 枚，BEE2_1S は 150×150 画像を 58,201 枚，BEE2_2S は 90×90 画像を 54,678 枚

で，(BEE)はミツバチが映っている画像，(NO-BEE)は映っていない画像である．

表 3.1　データセット数[21]

	Size	Train/Test (BEE)	Train/Test (NO-BEE)	Validate (BEE)	Validate (NO-BEE)	Total
BEE1	32×32	25,444	25,419	1,801	1,718	54,382
BEE2_1S	150×150	11,094	36,143	8,298	2,666	58,201
BEE2_2S	90×90	17,176	21,310	6,823	9,369	54,678

画像分類学習には ConvNet，VGG，ResNet，SVM を用い，それぞれで精度が良かったもの利用して 4 つの動画サンプルで評価実験を行っている．表 3.2 に各学習法とデータセットによる精度である．

表 3.2　学習分類機の精度[21]

	VGG16	ResNet32	ConvNetGS3	ConvNetGS4
BEE1	98.87%	99.48%	99.09%	98.63%
BEE2_1S	86.60%	68.90%	84.26%	86.36%
BEE2_2S	71.66%	73.73%	73.61%	75.56%

評価用の動画は，ミツバチの出入りが多い順に HT_Vid.mp4, MT_Vid.mp4, LT_Vid.mp4, NT_Vid.mp4 の 4 種類である．それぞれの画像分類機によるカウント数と，人の目視によるカウント数は表 3.3 のとおりある．

表 3.3　実験結果[21]

Video	Num.Frames	VGG16	ResNet32	ConvNetGS3	ConvNetGS4	Human Count
NT_Vid.mp4	742	151	75	182	127	73
LT_Vid.mp4	744	47	25	57	43	353
MT_Vid.mp4	743	1,245	145	597	316	2,924
HT_Vid.mp4	744	16,647	13,362	16,569	15,109	5,738

画像分類機のカウント数は，LT_Vid.mp4 と MT_Vid.mp4 で目視の半分以下で，1/10 以下の場合もある．MOG2 によって動体検知ができていないことや画像分類機の誤分類等が原因である．また HT_Vid.mp4 では目視の 2 倍以上となっており MOG2 が 1 匹のミツバチを複数の動体と誤検知したり，背景の草や葉，影等をミツバチとして誤検知してしまった結果である．このように極めて精度が悪く，動かないミツバチをカウントすらされないという問題もある．

- 高速度カメラによるミツバチの計数 [22]

吉田らは，図 3.9 に示した高速度カメラによるミツバチ計数システムを提案している．数百ヘルツで羽ばたく動作から得られる輝度の周期的な変化をフーリエ変換によって画素毎に調べ，飛翔するミツバチの軌跡を追跡する．巣箱周辺のおよそ 4m 四方の領域を解像度 1,024×1,024 フレーム数 2,000fps で撮影したとき，12~14mm サイズのミツバチを追跡できる．しかし，高速度カメラの動画を後で解析するシステムで，リアルタイム

の計測はできない．また，この実験で用いたスペックの高速度カメラは 200 万円程度と非常に高価である．さらに養蜂場に数十の巣箱があるが，それらを全てモニタして個体を識別することも不可能に近い．従って，あくまで研究のためのシステムであり，実用化を目的としたものではない．

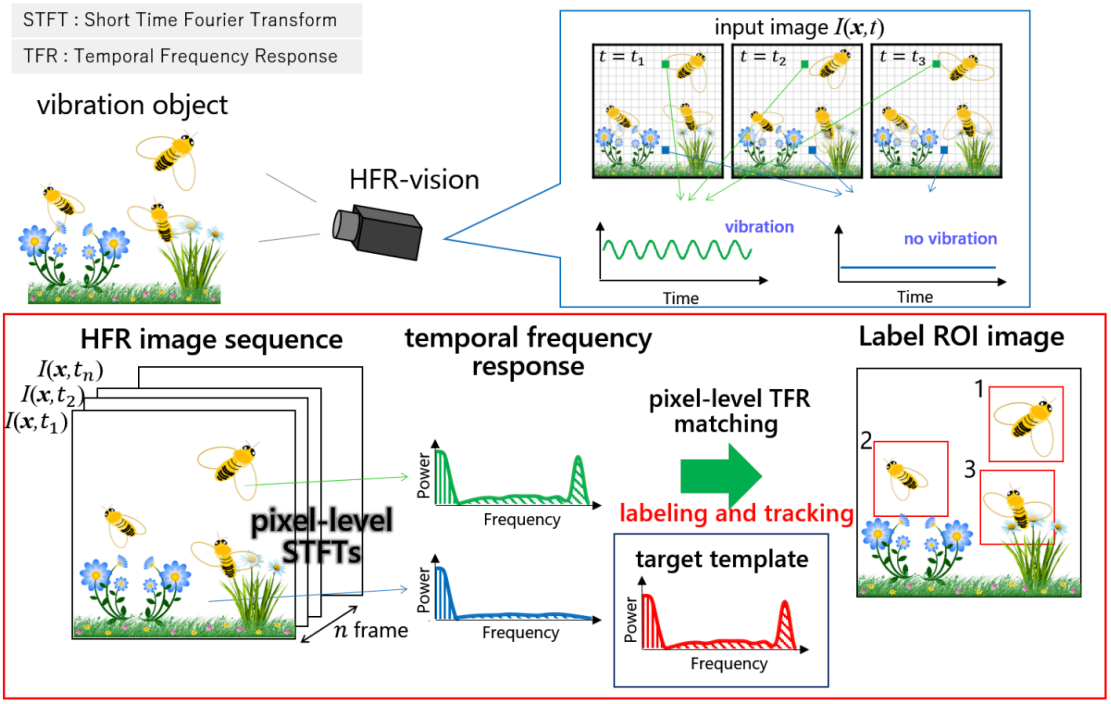


図 3.9　吉田らの手法[23]

● パターンマッチングによるミツバチの検出 [24]

島田は飛び立つミツバチと降り立つミツバチの検出を，フレーム間差分法と背景差分法，そして図 3.10 のように 2 値化した画像のテンプレートマッチングを用いて行なった．降り立つミツバチは比較的正確に識別できたが，飛び立つ場合は移動が速すぎるためか識別に失敗している．

図 3.10　島田の手法による 2 値化画像(左)とパターンマッチング画像(右)[24]

以上のように，IoT 技術を導入したミツバチの状態の把握，そしてカメラによるミツバチの個体数計測や動きの観察に関する研究はまだまだ発展途上と言える．

4　使用技術

4.1　マイコンボード

- Raspberry Pi [4]

Raspberry Pi Fundation が開発した ARM プロセッサを内蔵したシングルボードコンピュータである．SD カードを起動および記録用ストレージとして用いる．OS は主に Debian の派生ディストリビューションとして Raspberry Pi 用にカスタムした Raspbian が用いられ，Ubuntu や Windows10 IoT Core 等の OS も対応している．Raspberry Pi は大きく Raspberry Pi 1, 2, 3, 4，そして Zero の 5 つに分けられる．各モデルの諸元を表 4.1 に示す．Wi-Fi 機能を有したモデルは図 4.1 と図 4.2 の Raspberry Pi 4 Model B (以下 RP4 B と記す)，Raspberry Pi 3 Model B/B+ (以下 RP3 B/B+と記す)，Raspberry Pi Zero W (以下 RP Zero W と記す)の 4 種類あり，1,000~8,000 円程度で入手できる．本研究では主に RP4 B を用いてスズメバチ検出の実験を行う．また，比較対象として RP3 B と RP Zero W も用いた．

表 4.1　RP4 B, PP3 B/B+, RPZero W の諸元

モデル	RP4 B	RP3 B	RP3 B+	RPZero W
SoC	Broadcom BCM2711	Broadcom BCM2837	Broadcom BCM2837B0	Broadcom BCM2835
CPU	ARM Cortex-A72	ARM Cortex-A53		ARM1176JZF-S
	クアッドコア			シングルコア
	1.5 GHz	1.2 GHz	1.4 GHz	1.0 GHz
	64 ビット			32 ビット
GPU	Broadcom VideoCore VI	Broadcom VideoCore IV		
	500 MHz	400 MHz (3D 300 MHz)		250 MHz
		H.264/MPEG-4 AVC High Profile ハードウェアデコーダ・エンコーダ		
メモリ	4 GB	1GB		512MB
無線	IEEE 802.11ac 2.4GHz Bluetooth 5.0 BLE	IEEE 802.11ac 2.4GHz Bluetooth 4.2, BLE	IEEE 802.11b/g/n 2.4/5 GHz Bluetooth 4.1, BLE	IEEE 802.11b/g/n 2.4 GHz Bluetooth 4.1, BLE
電力	5V (最小 3A)	5V 400mA (2.0W)	5V 1.13A (5.661W)	5V 150mA (0.75W)
サイズ	5mm × 56mm	85.6mm × 56.5mm	85mm × 56mm	65mm × 30mm

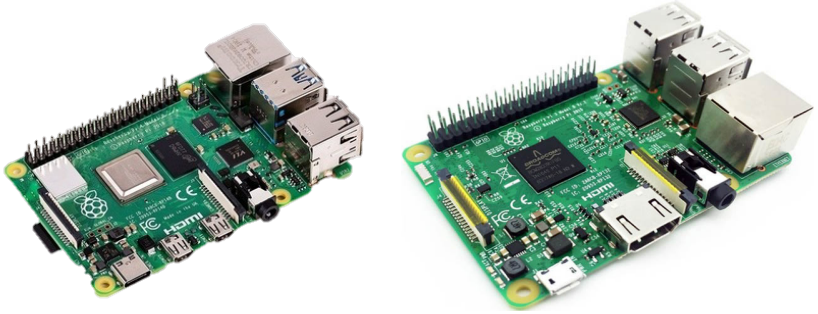


図 4.1　RP4 B(左)と RP3 B(右)

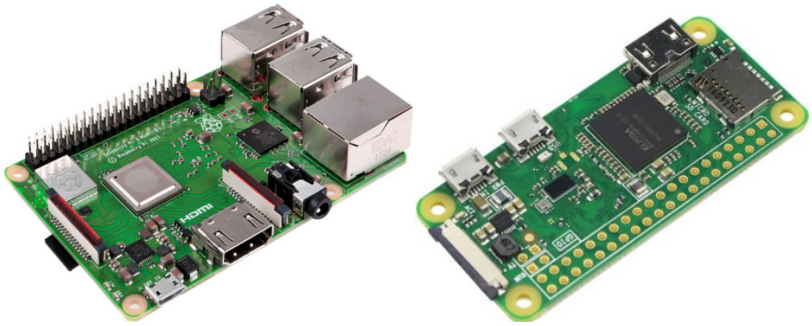


図 4.2　RP3 B+(左)と RP Zero W(右)

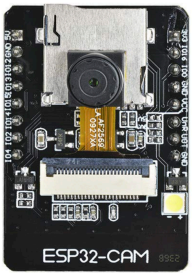
- Arduino [5]

ATMEL 社(現 Microchip 社)が開発した 8 ビット RISC マイコン AVR (Advanced Virtual RISCl)マイコン[25]を実装したワンボードマイコンである．ほぼ C++言語と互換の Arduino 言語とフリーの統合開発環境が利用でき，オープンハードウェアのため IoT 用ボードとして世界で広く利用され，数百円と安価な互換ボードや 32 ビットプロセッサを実装したボードなどが多数開発されている．Arduino の基本モデルは図 4.3 左の Arduino UNO であり，USB 経由で PC からプログラムや制御を行う．しかし IoT デバイスとして利用する場合，USB 接続ではなく Wi-Fi 等の無線通信が重要となる．純正の Arduino で適取得済かつ国内利用可能なのは, 図 4.3 中央の Arduino YUN だけであるが, 9,990 円と高価であり，また生産は既に終了している．

図 4.3　Arduino UNO (左)と Arduino YUN (中央)と WeMos D1 R32(右)

そこで，本研究では ESP-WROOM-32 (以下 ESP32) [26]を搭載した，図 4.3 の WeMos D1 R32 をセンサモジュールの通信に用いる．ESP32 シリーズは Espressif Systems Pte.(ESP)社が開発した 32 ビットマイコンで，Wi-Fi 802.11 b/g/n/e/i と Bluetooth v4.2 BR/EDR と BLE に対応している．TTGO T カメラ[27]や Ai-Thinker 社が開発した ESP32-CAM[28]等，ESP32 を搭載した複数のカメラモジュールも製品化されており，前者は 1500 円程度から，また後者は 800 円程度から購入可能である．なお，ESP32 にはいくつかのモデルがあり，技適を取得していないものもあるので，利用の際は注意が必要である．表 4.2 に両カメラモジュールの仕様を示す．

表 4.2　TTGO T カメラと ESP32-CAM の諸元

モデル	TTGO T-CAMERA	ESP32-CAM
外見		
サイズ	68.7×27.8×20.4mm	40.0×27.0×4.5cm
カメラモジュール	OV2640(1,600×1,400)	OV2640(1,600×1,400) OV7670(640×480)
マスターチップ	ESP32 32bit デュアルコア	ESP32-S32bit デュアルコア
Wi-Fi	802.11 b/g/n	802.11 b/g/n/d/e/i/k/r
Bluetooth	v4.2 BR/EDR	v4.2 BR/EDR

4.2　センサデバイス

● 重量センサ

力(質量，トルク)を検出するセンサで，ひずみによって電気抵抗が変わるロードセルに接続して使用する．図 4.4(左)の重量センサは一個 50 円程度と非常に安価なのに対して，ロードセルは一個 4~500 円で，台の 4 隅に使用した場合 2,000 円程してしまう．しかし，1,000 円程度で販売されている体重計にもロードセルが 4 つ入っており，これと筐体も利用し，ESP32 と HX711 を組み込めば非常に安価に無線重量計を制作することができる．図 4.4 右は実際に体重計に ESP32 と HX711 を組み込んだもので，図 4.5 はその回路構成（ホイートストンブリッジ回路）を示している．

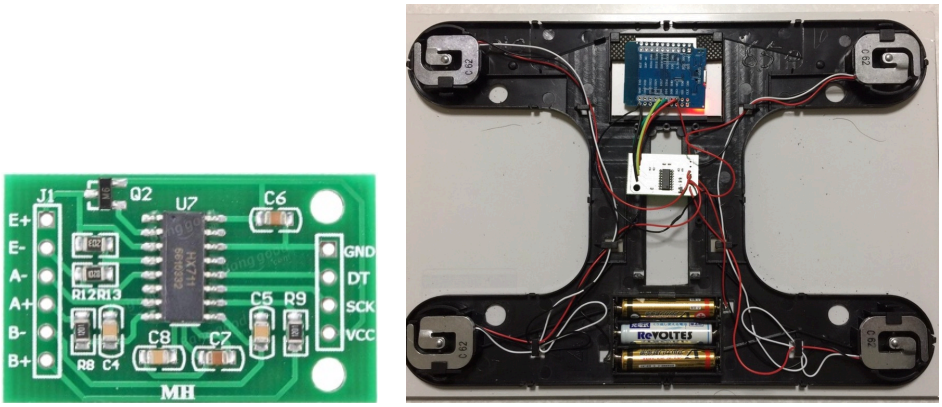


図 4.4　重量センサ HX711(左)とそれらを組み込んだ体重計(右)

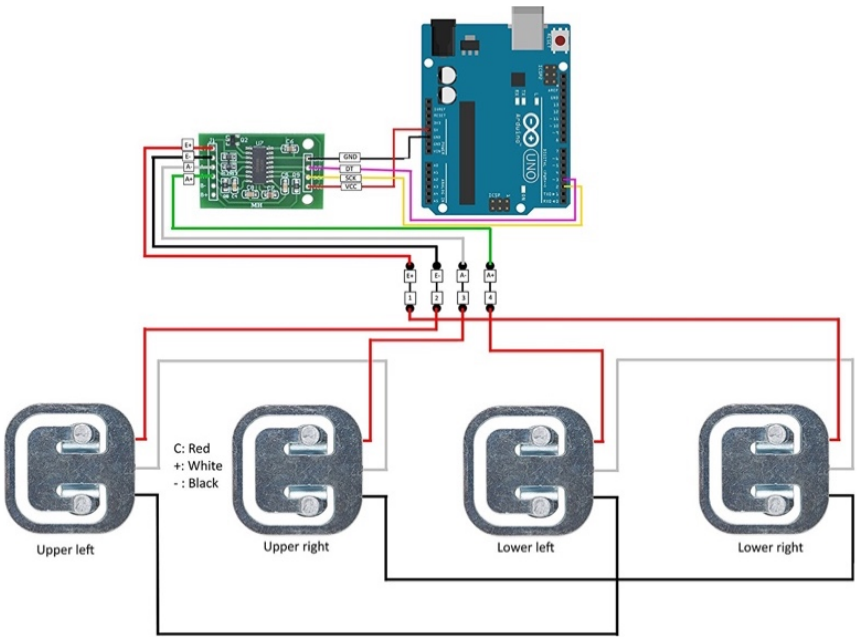


図 4.5　ホイートストンブリッジ回路

● 温湿度センサ

温湿度センサは安価で購入できる製品が多数存在する．本研究では表 4.3 の約 100 円の DHT11 と約 200 円の DHT21 を用いた．DTH11 の応答速度は遅く，計測のインターバルを 2 秒以上開ける必要があるとの記載があるが，安定してデータを取得するには，5V では 10 秒，3.3V で 15 秒程度開ける必要がある．DTH11 の精度が 1 度単位であるのに対して，DHT21 は 0.1 度単位となっている．測定間隔も DHT11 よりも短い．DHT21 は筐体が大きめであるが金属部分が覆われており，ケーブルも 2.5mm φ4 極オーディオケーブルのものなどがあり，単体での設置が便利なためミツバチの巣箱内部の温湿度計測に使用した．しかし，DHT シリーズは高湿度に長時間さらされると値が狂い，校正作業には恒温恒湿槽が必要となり使用不可能となる．巣箱が置かれる屋外は雨が降ると湿度は 100%に近くなり，実際に 1 ヶ月ほどで使用できなくなってしまった．

表 4.3　DHT11/DHT21

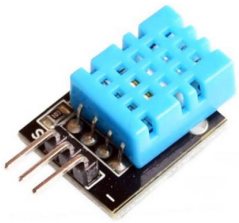
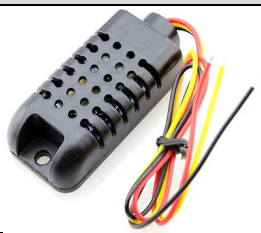
モデル	DHT11	DHT21
外見		
価格	0.7 ドル	2.1 ドル
温度測定範囲	0~50℃	−40~80℃
温度制度	±2℃	±0.5℃
湿度測定範囲 25℃	20~90%RH	0~100%RH
湿度制度	±5%RH	±3%RH

図 4.6 は Sparkfun と Adafruit の温湿度センサ HTU21D と HTU21D-F である．両者とも I_2C インタフェースで精度と基本的な使い方は同じであるが，後者は使用しているセンサモジュールの計測部分に保護フィルタ（-F は Filter の意味）があり，3.3V に加えて 5V 電源もサポートしている．精度と応答速度共に DHT センサよりも優れているが，保護カバーはないので利用時にはケースを用意する必要がある．価格は前者が中国のネット通販で 170 円程度と安価なのに対し，後者は 2,100 円程度と非常に高価である．また，図 4.7 は Si7021 温湿度センサで，赤が Sparkfun，青が Adafruit の製品で，両者の違いは Adafruit が 3.3V に加えて 5.5V をサポートしている点だけである．インタフェースは I_2C で HTU21D とアドレスも使い方も同じである．価格は Sparkfun が 800 円程度，Adafruit が 1,000 円程度である．より高性能の SHT31-D を搭載した図 4.8 の温湿度センサも販売されている．紫のセンサはメーカ不明であるが 300 円程度，青のセンサは 2,000 円程度である．以上から，Adafruit の HTU201D が最もコストパフォーマンスに優れており，今後の巣箱内センサにはこれを用いる予定である．

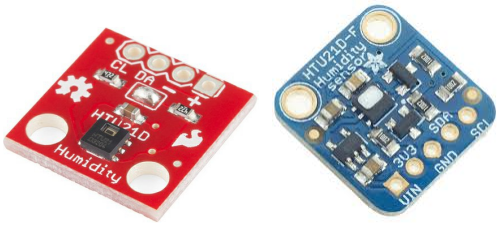


図 4.6　HTU21D (左)と HTU21D-F (右)

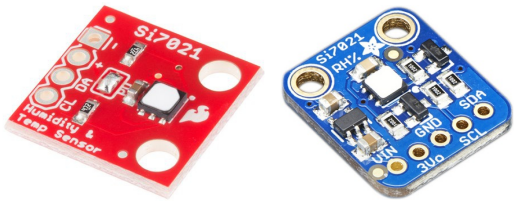


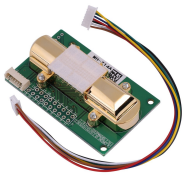
図 4.7　Si7021 赤 (左)と Si7021 青 (右)

図 4.8　SHT32 紫 (左)と SHT32 青 (右)

● 二酸化炭素センサ

巣内の二酸化炭素濃度測定には，安価で測定範囲も広い MH-Z19B を本研究では用いた．他の製品との比較を表 4.5 に示す．

表 4.5　二酸化炭素センサ比較表

モデル	MH-Z19B	MH-14A	CCS-811	MG811
外見				
価格	1,600 円程度	1,800 円程度	2,700 円程度	2,600 円程度
測定範囲	0~10,000ppm	0~5,000ppm	400~8,192 ppm	0~10,000ppm
起動時間	3 分	3 分	20 分	
検出技術	赤外線吸収ガス	赤外線吸収ガス		固体電解質

4.3 エッジ AI デバイス

Raspberry Pi の USB ポートに接続することで，学習モデルに基づく推論を高速化するアクセラレータが製品化されている．図 4.9 左は Google の Coral USB Accelerator [29]，右は Intel の NCS2 (Neural Compute Stick 2) [30]である．

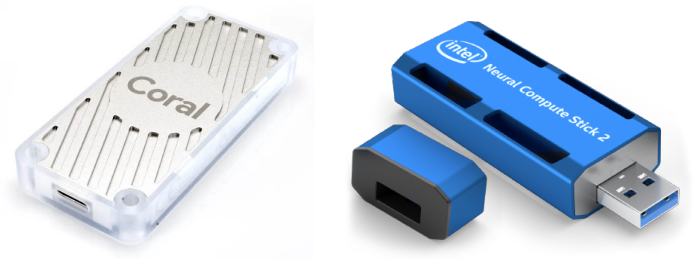


図 4.9　Coral USB Accelerator (左)と NCS2 (右)

Coral USB Accelerator は機械学習に特化した Edge TPU (Tensor Processing Unit)を搭載し，Raspberry Pi などの Debian 系 Linux 上で動作する．AI プログラミングの外部演算装置として動作し，また，Google が提供する様々な機械学習モデルを利用することが可能である．本研究のスズメバチ検出では，Raspberry Pi 上で TensorFlow Lite を利用している．TensorFlow Lite は Google が開発した機械学習のオープンソースソフトウェアライブリ TensorFlow のモバイルデバイス版である．Coral USB Accelerator は TensorFlow Lite をサポートしているので，本研究の実験にこのデバイスを用いている．

NCS2 は VPU(Vision Processing Unit) という半精度浮動小数点数と 8bit 固定小数点数の行列演算ユニットを有し，機械学習の推論処理をこの行列演算ユニットで行う．Intel が提供する AI 用 SDK の OpenVINO (Open Visual Inference and Neural network) における推論処理を高速化する．TensorFlow で生成した学習モデルも OpenVINO 用に変換して利用することができる．しかし本研究は TensorFlow ベースで進めており，あえて OpenVINO を導入する明確なメリットがないことから，今回は利用を見送っている．

4.4 機械学習による物体検出

機械学習による物体検出は，R-CNN (Regions with Convolutional Neural Networks)，YOLO(You Only Look Once)，SSD (Single Shot multibox Detector)の 3 つの系統のアルゴリズムに大別される．以下で，それぞれについて簡単に説明する．

- R-CNN 系

R-CNN [31]は画像検出手法の基本である畳み込みニューラルネットワーク(CNN)の演算量を減らすため，あらかじめ物体の候補領域 (Region Proposals)を Selective Search で絞り込んで CNN を行うものである．図 4.10 に概略を示す．入力画像から候補領域を最大で 2,000 個切り出し，それらを全て固定サイズに変換した後，CNN にかけて特徴量

(features)を抽出する．その特徴量を用いて候補領域を SVM (Support Vector Machine)で分類し，どの物体（あるいは背景）なのかを判定し，物体の Bounding Box を推定する．R-CNN の提案当時の既存手法の精度を 30%以上改善したが，まだ大きな演算量を要し，GPU を使って 10~45 fps と遅い欠点がある．

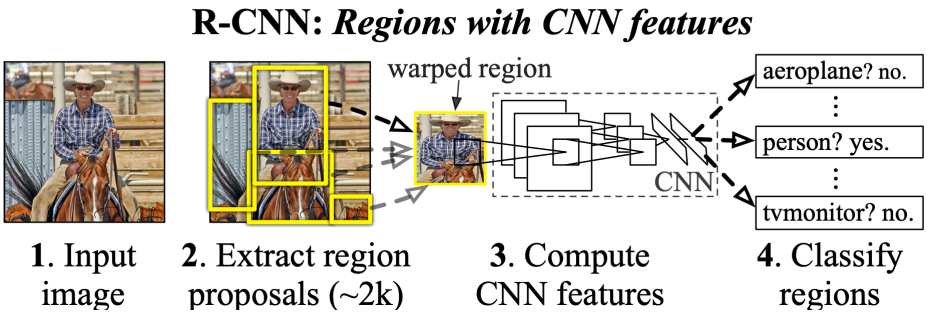


図 4.10　R-CCN [31]

Fast R-CNN [32]は図 4.11 のように，関心領域(RoI: Region of Interest)を max pooling によって固定サイズの領域に圧縮する．なお，RoI は R-CNN の候補領域(Region Proposals)と同意である．これによって，R-CNN の最大 2,000 回の CNN 処理が 1 回で済む．また，CNN，SVM，Bounding Box 回帰子(regressor)の 3 種類を単一ネットワークで計算する．この 2 つの改善により，実行速度は 16 層のニューラルネットワーク VGG-16 を用いた R-CNN より 9 倍の学習速度，213 倍の識別速度を達成している．

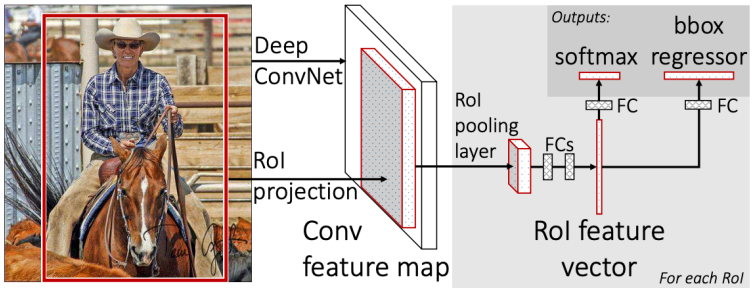


図 4.11　Fast R-CCN [32]

Faster R-CNN [33]は図 4.12 に示すように，Fast R-CNN のボトルネックである Selective Search による RoI 生成部分を RPN(Region Proposal Network) と言われる CNN 構造の小さな畳み込みネットワークに入れ替えた点が大きな特徴である．これにより入力と出力の関係を直接単一のモデルで学習することに初めて成功した．また，Anchor と呼ばれる検出矩形パターンを複数用意し，Anchor 毎に物体である可能性が高いものだけを RoI プーリング以降で処理するよう RPN に学習させる．これらにより，Faster R-CNN は Fast R-CNN より 10 倍の速度を達成している．

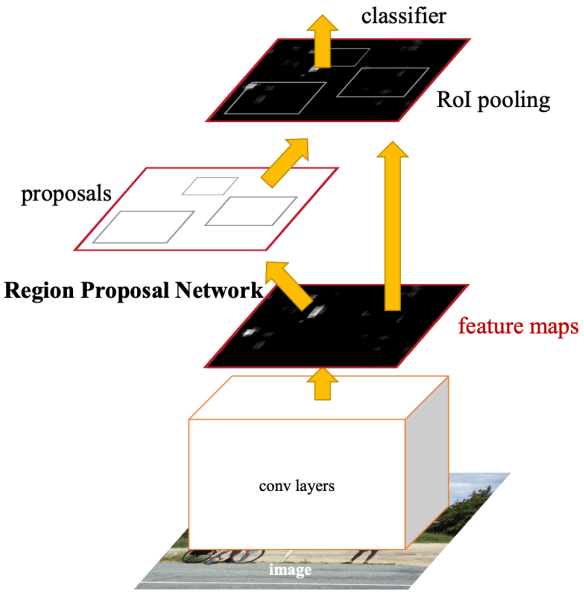


図 4.12　Faster R-CCN [33]

- YOLO [34]

R-CNN は画像の領域推定と分類が分かれているが，YOLO は画像全体から直接，物体の検出と分類を一度に行うことで高速化を図っている．画像全体をグリッド分割し，その領域ごとに物体のクラスと Bounding Box を求める．画像全体を処理することで背景の誤検出は Fast R-CNN の約半分に抑えられる．しかし，小さい物体の検出やグリッド内に大量の物体がある場合が苦手である．識別精度は Faster R-CNN にやや劣るが，Titan X GPU を用いた実験で 45fps，高速実装で 155fps を記録し，Faster R-CNN のおよそ 6~7 倍の高速化を実現している．YOLOv2，YOLOv3 と精度を向上させており，本研究で使用している物体検出ライブラリ ImageAI のアルゴリズムに採用されている．

- SSD [35]

SSD は YOLO と同様に入力画像から直接 CNN で物体を検出する．YOLO は出力層だけで Bounding Box を生成していたが，SSD は各層の大きいな特徴マップも利用することで小さな物体も検出可能である．そのため比較的低解像度の画像でも高い精度が得られ，高速化に向いている．300×300 の画像サイズで 59fps を達成している．

4.5 物体検出ライブラリ

本研究でスズメバチの検出に，Python ライブラリである下記の ImageAI[36]，TensorFlow[37]，TensorFlow Lite[38]を用いた．

- ImageAI[36]

数行のコードで自己完結型の深層学習およびコンピュータービジョン機能を備えたアプリケーションやシステムが構築できる．予測クラスと検出クラスが用意されており，

予測クラスでは SqueezeNet，ResNet，InceptionV3，DenceNet の 4 つの学習済みモデルが，検出クラスでは YOLOv3，TinyYOLOv3，RetinaNet の 3 つの学習済みモデルが 10 行程度の Python プログラムで実行可能である. ユーザは学習データ 200 枚以上と Pascal VOC 形式でアノテーションした xml ファイルを用意するだけで，YOLOv3 の事前学習モデルから転移学習を行ってオリジナルの学習モデルを生成して使用可能である.

- TensorFlow[37]

Google が開発した機械学習用オープンソフトウェアライブラリである．64 ビットの Linux，macOS，Windows 上で動作し，プログラミング言語は C，C++，Python，Java，Go に，ハードウェアは CPU，NVIDIA GPU，Google TPU に対応している．クラウドやオンプレミスをブラウザから，また PC やモバイルデバイス上など，柔軟かつスケーラブルな実行環境が利用できる．本研究では，GPU パソコン上でスズメバチの学習に TensorFlow の Object Detection API を用いた.

- TensorFlow Lite[38]

計算資源の限られたモバイルやマイコンボード上で推論を高速に行うためのツール&ランタイムライブラリ群で，TensorFlow で生成した学習モデルを“.tflite”ファイルに変換して使用する．モデルの重みを 4 バイト float からバイト整数に変換することでサイズを 1/4 にし，最大で 2~4 倍に高速化され，かつ精度低下は 1%未満であるという．Google Edge TPU は 8 ビットのみサポートのため, “.tflite”形式のみに対応している. サーバに接続せずにローカルで実行できるため, ネットワークによるデータ送受信の遅延が生じず，データの漏洩の心配がなく省電力な AI エッジコンピューティングが実現できる．本研究では Raspberry Pi 上で物体検出を行う際に，TensorFlow Lite を用いた.

5　センサシステムの開発

5.1　概要

本研究の目的は，ミツバチの異常や外敵の検出を行うために様々なセンサやカメラを養蜂施設に設置し，また風雨にさらされネットワークが不安定な屋外での安定動作の可能性を検証することである．そこで，TBS の「みつばちプロジェクト」[9]として本社社屋で 6 群の蜜蜂を飼育している養蜂施設「赤坂みつばちあ」と，原宿竹下口横の商業施設「はらじゅくアッシュ」屋上で同じく 6 群を飼育している「BEETOPIA はらじゅく」[10]にご協力いただいた．「赤坂みつばちあ」では図 5.1 左のように本研究室が設置したビデオカメラを，無線で基幹システムに接続しており，図 5.1 右のように CSR 活動の一環として TBS の Web サイトで動画を 24 時間ライブ配信している．

図 5.1　赤坂みつばちあのライブ配信

養蜂の経験がゼロであり，まずミツバチの生態について勉強する必要があったため，2019 年 2 月から「赤坂みつばちあ」の活動に参加した．また，ミツバチの異常がどのようなもので何のセンサでそれが検知できるかも不明なため，外気温湿度，巣箱内温湿度，重量，二酸化炭素の各センサと赤外線カメラを設置し，LTE モバイルルータでデータをサーバにアップロードした．以下では各センサについて説明を行う．

5.2　重量・温湿度センサシステム

図 5.2 にシステムの概要を示す．市販の体重計 [39]の筐体と，4 つのロードセルを利用し，24 ビット AD コンバータを搭載した重量センサ HX711 で計測したデータは I_2C 通信で ESP32 を実装したマイコンボード WeMos D1 R32 に送られる．また巣箱内と外気の温湿度はそれぞれ表 4.3 に示した温湿度センサ DHT21 と DHT11 で計測し，これも WeMo sD1 R32 に送られる．そこから LTE モバイルルータを経由して，リアルタイム性に優れた IoT 用通信プロトコル MQTT で研究室の Raspberry Pi サーバに転送される．Raspberry Pi に実装したフローベースのプログラミングフレームワーク Node-RED を用

いて，グラフ表示とデータの保存を行う．重量・温湿度センサモジュールの部材費は2,000 円程度である．

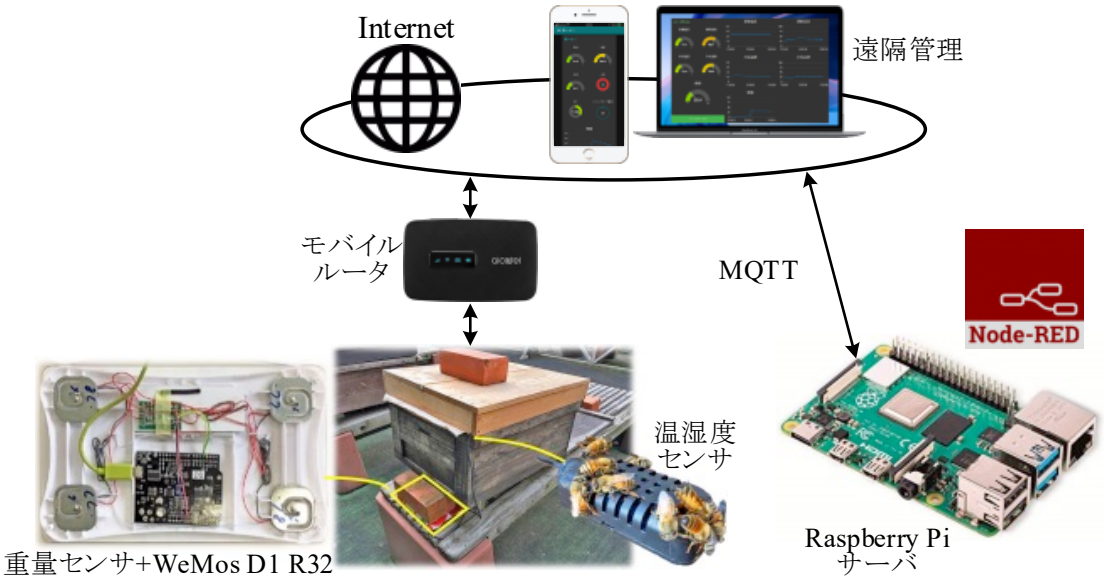


図 5.2　重量・温湿度システム

6 群の巣箱の内の 1 つの下に重量計を設置し，その裏に外気の温湿度を測定するセンサ DHT11 を取り付けた．また巣箱の蓋の端に溝を掘って DHT21 のコードを内部に引き込み巣枠の間に垂らすようにした．2019 年 7 月 5 日の巣箱重量，温度，湿度をそれぞれ図 5.3~5.5 に示す．降水量は気象庁の東京気象観測地点のデータ[40]を参照した．

この日は雨のためミツバチの出入りはほとんどないが，図 5.3 の重量は 7 時頃から次第に増加している．これは雨が降りはじめ，それを木製の巣箱が吸ったためである．ミツバチは巣箱内の温湿度を一定に保とうとし，暑いときは巣房に溜めた水を扇風して気化熱で温度を下げ，寒いときは筋肉を動かすことで発熱して温度を上げる．図 5.4 から外気温は 25℃を下回っているが巣箱内は 32~35℃を保たれ，また図 5.5 から外気湿度は 90%以上に対して巣箱内は 70~75%となっていることがわかる．

図 5.6 は 2019 年 8 月 5~12 日の 1 週間のデータである．この期間はずっと天気が良く雨は降っていない．真夏でも巣箱内の温度は 36 度前後，湿度は 70%前後に保たれている．重量は日の出後にミツバチが巣を飛び出していく朝 5 時頃の重量が最も軽く，蜜を集めたミツバチが全て戻る夕方が最も重い．その後，巣の中で貯蜜のために羽で扇いで水分を飛ばすため明け方が一番軽くなる．採蜜期の 4~5 月は，上下変動を繰り返しながら重量は右肩上がりに増えていくが，6~9 月は密の出る花がほとんどなく，8 月には群のミツバチの数もピークを過ぎるため，次第に重量が減っていることがわかる．なお，8 月 7 日に縦線が出ているが，これは巣箱の内見作業で蓋の開け閉めや二段になっている巣箱の上段を持ち上げるなどをしたためである．

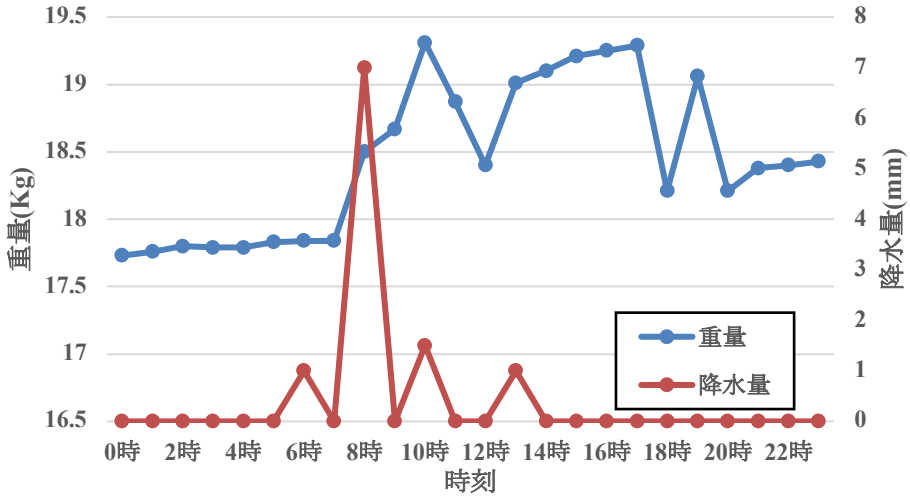


図 5.3　一日の巣箱重量変化と降水量

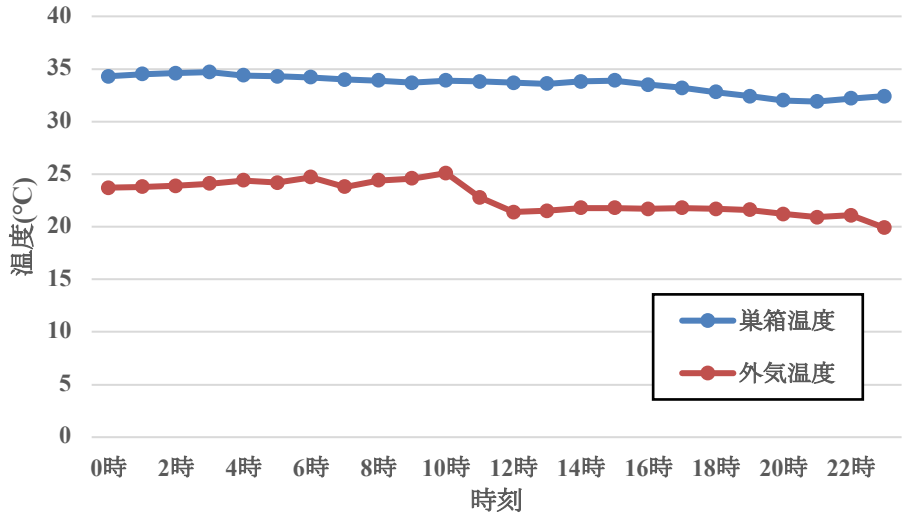


図 5.4　一日の温度変化

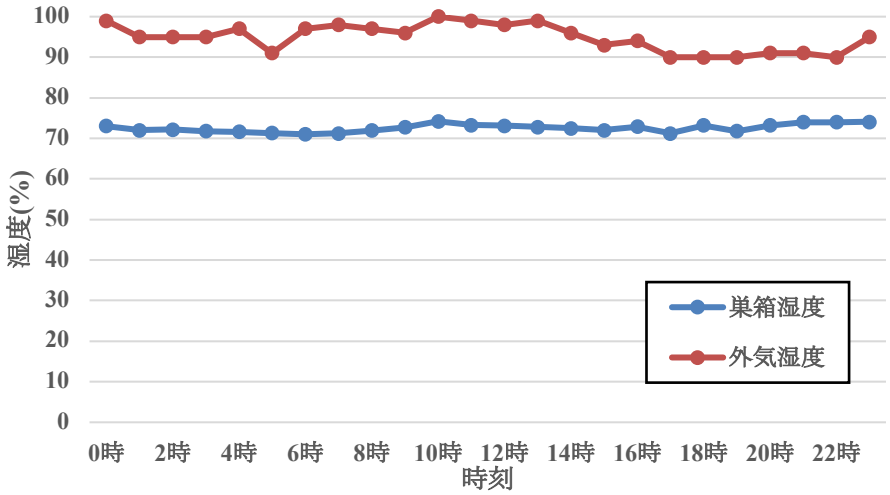


図 5.5　一日の湿度変化

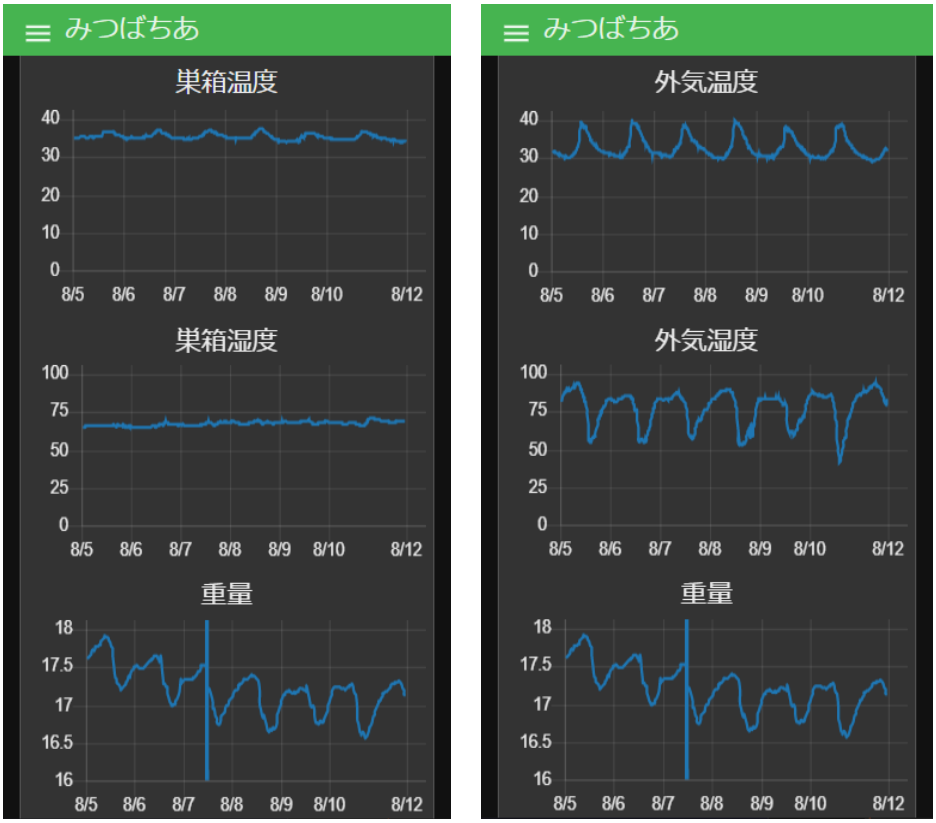


図 5.6　8 月の一週間の計測データ

今回は試験的に 1 群のみにセンサを設置しており，そこではダニ等の異常は発生せず，また分蜂を起こさないように手入れをしている．そのため巣箱の温湿度は一定に保たれ正常であることは把握できたが，3.1 節で紹介した製品 Bee Sensing のように，温湿度でダニや分蜂などの異常検知が可能かどうかを調べることはできなかった．しかし，頻繁に内検しなくとも，遠隔モニタで温湿度が正常であり，また重量変化から蜂の群勢がわかるなどの有用性が示された．また 4~5 月は，放置していると蜜がいっぱいになり，大きな群は分蜂を起こしやすくなるので，毎週採蜜を行う必要がある．また，ハチの個体が増えた場合は巣箱を二段にする作業も行う．重量センサによりいつ採蜜作業を行えばよいかが判断でき，大変有用であったという評価をいただいている．

そして，半年の屋外での使用において基本的にシステム自身が不具合をおこすことはなく，安定性と信頼性が確認された．ただし，外気測定用の DHT11 は仕様書にも記載されているが，高湿度に長時間さらされると正しく計測できなくなるという問題があり（DHT21 も同様），実際に外気の湿度測定ができなくなってしまった．DHT11/21 は IoT 用温湿度センサとして最も広く使われている製品であるが，4.2 節にも示したように，このほかにも多くの温湿度センサが販売されているので，それらの性能も検証しているところである．また，作業中にセンサのコードが引き抜かれたり，フタに掘った溝にコードを通さずに無理矢理フタを締めて断線するなどのトラブルがあった．そこで作業の邪魔とならないように，バッテリー駆動によるセンサモジュールを開発中である．取得し

たデータは Node-RED ダッシュボードにリアルタイムで表示されるが，指定した期間を過ぎるとデータが消えてしまうため，mySQL データベースに保存していた．一つ一つのデータは小さいものの，時間と共に数が膨大となりもともと事務処理等に向いている関係データベース（RDB: Relational DataBase）の mySQL では処理することが極めて困難となってしまった．そこで，時系列データベース（TSDB: Time Series DataBase)の一つである InfluxDB でデータベースを再構築中である．

5.3　二酸化炭素濃度計測

ミツバチは羽を使って巣内を換気することで，温度や湿度と同様に二酸化炭素濃度も一定に保っている [41]．そこで二酸化炭素センサ MH-Z19B を RP3 B に接続し，1 分間隔で計測を行った．測定期間は忙しい時期を過ぎて作業の邪魔にならない，2019 年 9 月 17 日 18 時~19 日 6 時までの 1 日半である．そのうちの 18 日 0~24 時の 1 日のデータを図 5.7 に示す．気象庁の発表による岩手県大船渡市綾里の 2019 年 9 月の平均二酸化炭素濃度は 407.1ppm であった[42]．東京の平均はもっと高い値を示すと思われるが，気象庁の公開データは，綾里，南鳥島，与那国島の三ケ所だけである．国の建築物環境衛生管理基準では，室内の二酸化炭素濃度を 1,000ppm 以下に抑えるよう定められており，それを超えると思考力や集中力が低下すると言われている．図 5.7 の正常であった巣箱内の濃度は，およそ 400~700ppm で推移している．異常時にどのような変化があるかは現在のところ不明であるが，このデータから 800ppm を超え続けると問題が発生していると考えられる．二酸化炭素濃度については，今後の検討課題である．

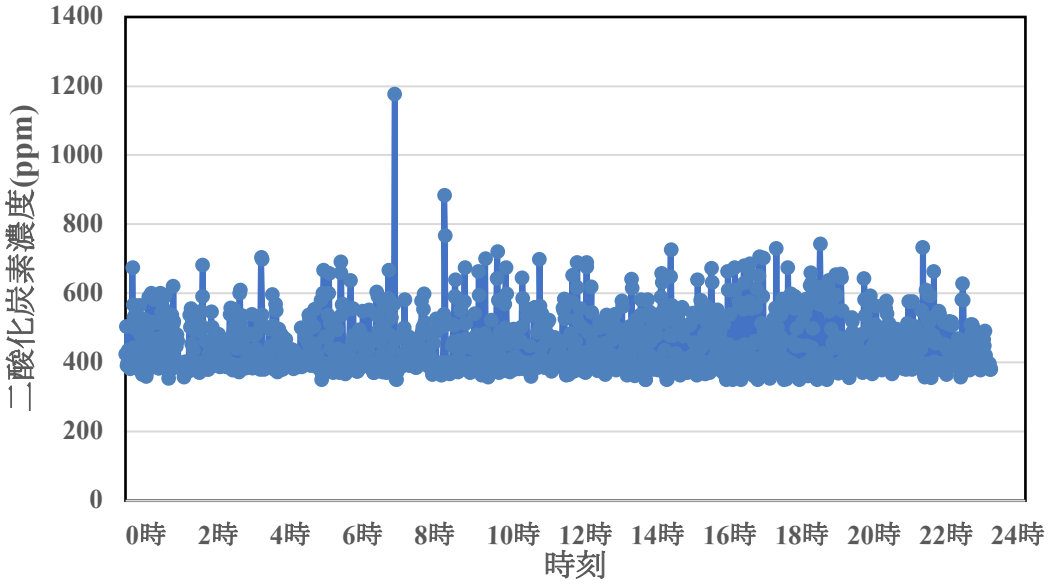


図 5.7　二酸化炭素濃度センサ

5.4　巣箱内カメラシステム

養蜂作業において巣箱の内見作業はミツバチ群の健康状態を知る上で，非常に有効である．また，ミツバチは採蜜を日中に行い，夜は休むことはあっても寝ることはなく，巣箱の中で 24 時間働き続けている．そのため，巣箱内のミツバチの様子をモニタすることで有益な情報が得られると考え，リアルタイムの配信システムを構築した．

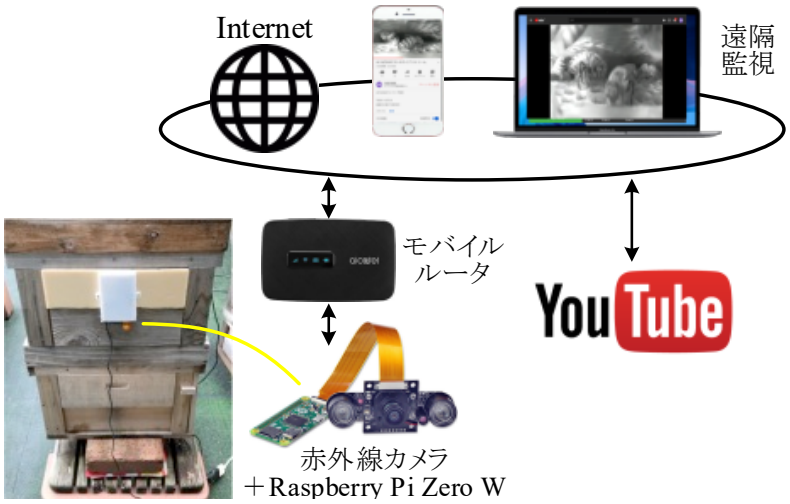


図 5.8　巣箱内カメラのシステム概要

図 5.8 にシステムの概要を示す．WiFi 機能を有した小型の RPZero W のカメラポートに専用の赤外線カメラを接続し mp4 形式に圧縮した動画データを，LTE モバイルルータを経由して，Youtube の無料ライブ配信サービスに転送した．ライブ配信には YouTube のライブ配信機能を用いた．動画はライブラリ raspivid と ffmpeg を用いて，Full-HD (1920×1080)の解像度でフレームレートは 30fps に圧縮した．これは下記のようなコマンドで簡単に実行することができる．

```
raspivid -o - -t 0 -rot $ROT -fps 30 -b 4500000 | ¥
ffmpeg -loglevel warning -re -f aac -i $AFIFO -r 30 -f h264 ¥
   -i - -vcodec copy -acodec copy -bsf:a aac_adtstoasc ¥
   -f flv rtmp://a.rtmp.youtube.com/live2/$STREAM_KEY
```

巣箱の裏には換気用の金網の窓があり，ここに固定するケースを 3D プリンタで制作して図 5.9 左のように設置した．図 5.9 右が巣箱内の映像であるが，実際の向きは写真左が地面のほうの下側で写真右が上側である．ミツバチの視覚の三原色は，黄色，青，紫外で，赤は感知できない．そのため，赤外線カメラに付属の赤外線ライトを照らしても活動に影響は見られなかった．なおライトを照らさず，ミツバチの体温だけではほとんど何も映らなかった．巣房の掃除や造巣の様子を観察することができたが，一台のカメラで撮影できるのは 10 枚の巣枠の間の 1 ヶ所だけであり，どこにカメラを仕掛けれ

ばより有益な情報が得られるかは今後の検討課題である．

なおミツバチは夜間に採蜜は行わないが，巣門前では巣内の掃除で出たごみや死骸を外に出す等の活動を続けている．そこで，赤外線カメラで巣門を上から撮影することも試しており，この映像を用いたミツバチの個体数カウントの実験も行った．これについてはまた 7 章で解説する．

図 5.9　赤外線カメラの設置(左)とライブ配信(右)

6　スズメバチ検出システム

6.1　概要

巣箱にカメラを設置して機械学習による推論でスズメバチの襲来を検知し，リアルタイムで利用者に伝えるシステムを構築するために，学習データの収集と各種物体検出アルゴリズムの性能評価，そして複数のカメラを Wi-Fi で Raspberry Pi サーバに接続するエッジコンピューティングシステムの実装を行った．スズメバチの学習データは Web から写真を集めるとともに，養蜂施設にカメラを設置して動画を撮影した．それらのデータから 3 つのデータセットを生成し，YOLO，Faster R-CNN，SSD に対して学習させることで各データモデルの検出精度の評価を行う．また，TensoFlow Lite に対応した SSD による処理速度を Raspberru Pi 上で評価する．

6.2　スズメバチの動画撮影

東京湾アクアラインを越えた先の千葉県袖ケ浦市の住宅街の一角に，株式会社坊ノ内[43]の養蜂施設がある．図 6.1 に示したように，20 群の養蜂箱の 1 つに対して正面，斜め，横の 3 台のネットワークカメラ SpotCam[44]を設置し，スズメバチの活動が活発な 2019 年 10 月 24 日~11 月 4 日の 11 日間撮影を行った．SpotCam は台風にも耐えられる防塵・防水規格 IP65 を満たし，赤外線による夜間の撮影も行うことができる．また，1 日分のクラウド録画の無料サービスを利用し，動画の中からスズメバチが映っている部分を目視で調べてダウンロードし，機械学習に使用した．カメラの解像度は正面と横に設置した SpotCam-HD-Pro が HD (1,280×720)，斜めの SpotCam-Sense-Pro が FHD (1,920×1,080)で，エンコードはいずれも最大フレームレート 30fps の H.264 である．各カメラで撮影したサンプル画像を図 6.2~6.4 に示す．

図 6.1　袖ケ浦市の養蜂場に設置したネットワークカメラ

図 6.2　正面からの映像 (SpotCam-HD-Pro)

図 6.3　横からの映像 (SpotCam-HD-Pro)

図 6.4　斜めからの映像 (SpotCam-Sense-Pro)

表 6.1 にカメラの設置期間で，スズメバチが巣箱前に滞在した時間と個体数を 3 時間ごとの合計で示す．スズメバチは主に朝方から昼頃にかけてミツバチを襲いに来ており，数分間滞在することが多い．また 11 月 1 日には同じ個体が 20 分も襲い続けるシーンもあった．しかし，夜間や雨天時にはスズメバチが襲来することはなかった．

表 6.1　スズメバチの滞在時間と個体数

天候	晴れ	曇り	晴れ	雨	晴れ	曇り	晴れ	晴れ	曇り	晴れ
日付	10 月 26 日	27 日	28 日	29 日	30 日	31 日	11 月 1 日	2 日	3 日	4 日
6:00~9:00	100s (1 匹)	120s (2 匹)					480s (1 匹)			900m (1 匹)
9:00~12:00	20s (1 匹)	60s (1 匹)			10s (1 匹)		300s (3 匹)	60s (1 匹)		
12:00~15:00							1,200s (2 匹)			
15:00~18:00							1,260s (1 匹)			

6.3　学習用データセットの作成

スズメバチ検出の機械学習では，以下に示す 3 種類のデータセットを用いた．

- データセット 1

500 枚のラベル付きの画像である．500 枚の内訳はインターネット上から収集した 380 枚と養蜂施設で撮影した動画から生成した 120 枚である．インターネット上の画像収集には Google, Bing などの画像検索サービスから自動的に画像をダウンロードする Python のパッケージ icrawler を用いた．また，養蜂施設の 120 枚は正面から撮影した動画からスズメバチが映っているフレームを切り出して生成した．機械学習において，1 枚あたりの画像サイズが大きいと学習時間が長くなるため，収集した 500 枚を 320×320 に縮

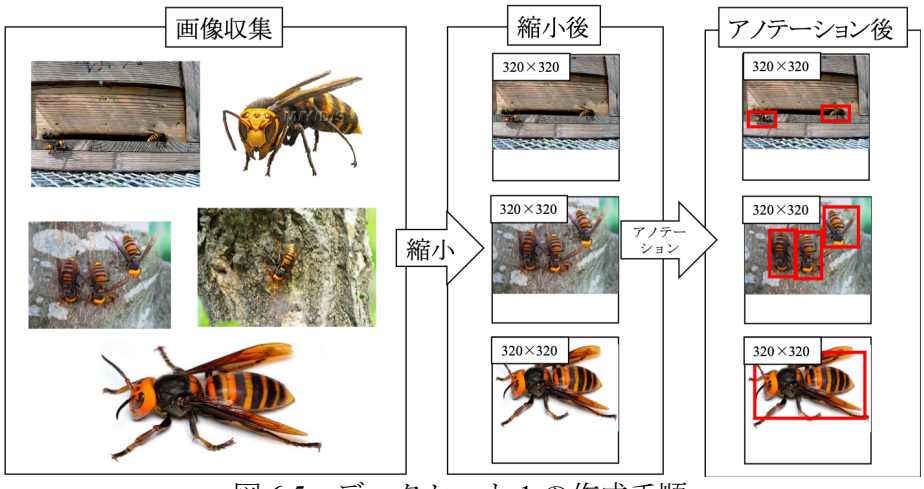


図 6.5　データセット 1 の作成手順

小した．縦横比が 1:1 でないものは縮小で空いた場所を白色で塗りつぶしている．アノテーション作業には labelImg [45]を用い，矩形を Pascal VOC 形式で xml ファイルに出力した．後のアノテーション作業も同様に行なった．図 6.5 に収集したデータからデータセット 1 を作成する手順を，図 6.6 にそのデータセットの一部を例示する．

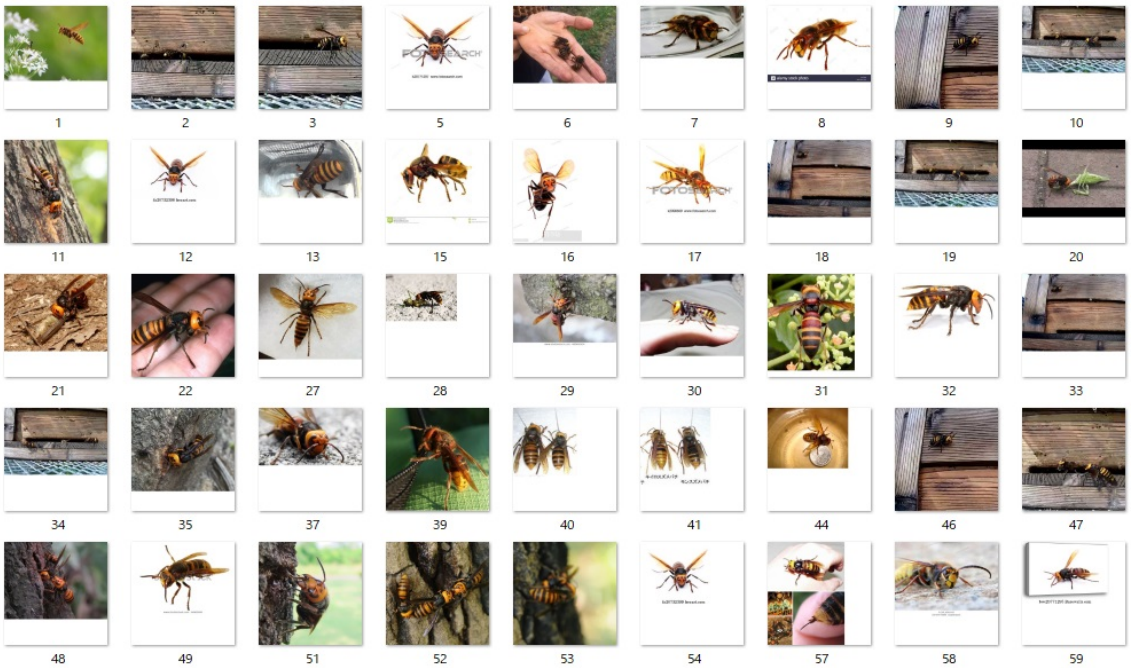


図 6.6　データセット 1 の画像例

● データセット 2

データセット 2 は 1,000 枚のラベル付きの画像である．1,000 枚の内訳はデータセット 1 の 500 枚に，養蜂施設の横と斜めを含む動画から生成した画像 500 枚である．図 6.7 にデータセット 2 の作成手順を示す．横と斜めの動画はスズメバチのサイズが正面に比べてかなり小さいため，このまま縮小すると特徴量が減ってしまう．そこで元画像

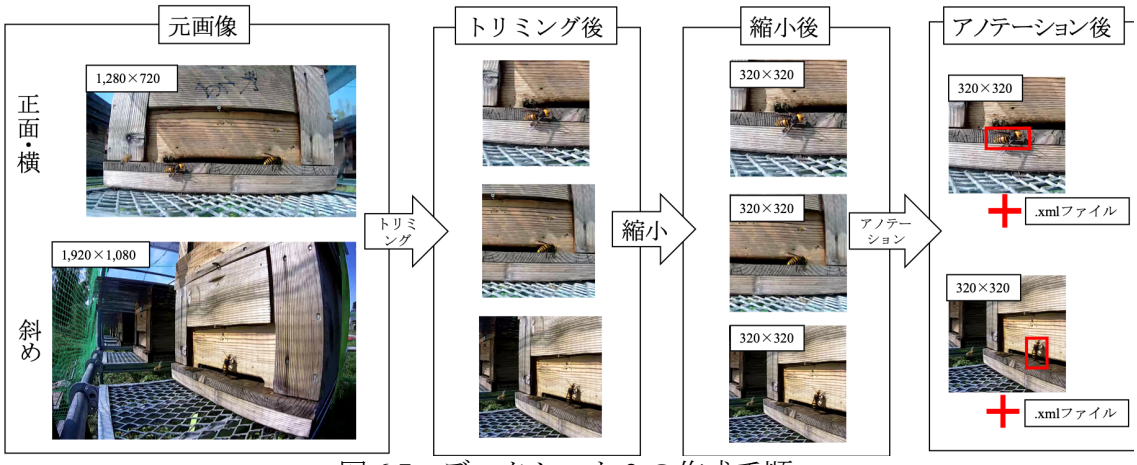


図 6.7　データセット 2 の作成手順

をスズメバチが映るように 320×320 以上の正方形でトリミングをした後に 320×320 に縮小し，アノテーションを行なった．図 6.8 にデータセット 2 の一部を示す．

図 6.8　データセット 2 の画像例

● データセット 3

データセット 3 は 1,400 枚のラベル付き画像である．1,400 枚の内訳はデータセット 2 の 1,000 枚に加えて，養蜂施設の動画から生成した画像 400 枚である．図 6.9 にデータセット 3 の作成手順を示す．元画像をトリミングせずに正面と横から撮影した画像は 640×360，斜めから撮影した画像は 480×270 に縮小し，アノテーションを行なった．図 6.10 にデータセット 3 の一部を示す．

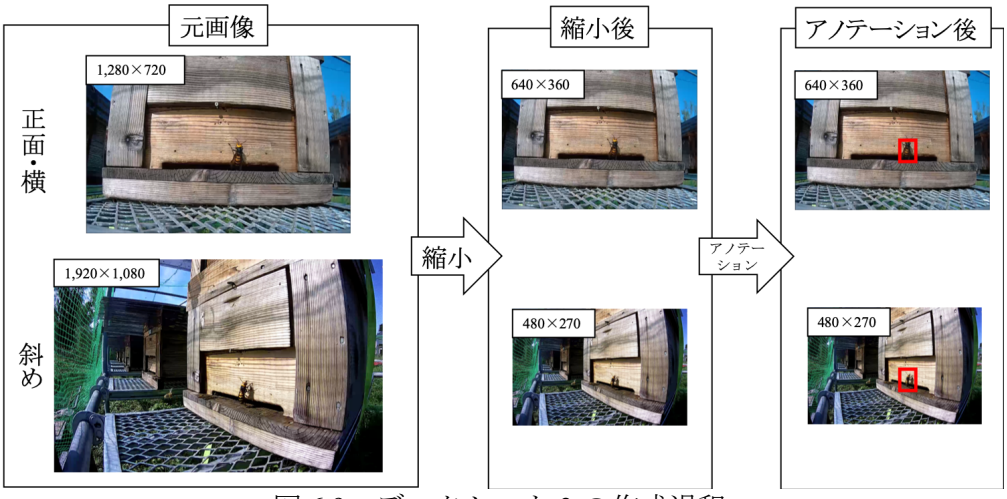


図 6.9　データセット 3 の作成過程

図 6.10　データセット 3 の画像例

6.4　機械学習によるスズメバチ検出

データセット 1 を用いて，ImageAI は Intel Core i7 パソコンで，TensorFlow は GPGPU Geforce GTX 980Ti を搭載した Core i5 パソコンで機械学習を行った．表 6.2 に実験環境をまとめる．

表 6.2　機械学習の実験環境

	ImageAI	TensorFlow		
OS	Windows 10 Home	Windows 10 Home		
Mother Board		MSI H97		
CPU	Intel Core i7−4770 @ 3.4GHz 4 cores	Intel Core i5-4590 @ 3.30GHz 4 cores		
Memory	16GB	16GB		
GPU	なし	GeForce GTX 980Ti		
		詳細	CUDA コア	2,816
			ベースクロック	1GHz
			消費電力	250W
			バスタイプ	PCI-E 3.0
			DirectX	12 API
			OpenGL	-
			メモリスペック	
			メモリクロック	7.0Gbps
			メモリ量	6.14GB
			メモリ I/F	384-bit GDDR5
			最大バンド幅	336.5 GB/s

精度評価には，表 6.3 の 6 つの動画ファイルを用いた．hornet1.mp4 と hornet2.mp4 は養蜂場で撮影した正面の動画，hornet3.mp4 と hornet4.mp4 は斜めの動画，hornet5.mp4 と hornet6.mp は横の動画である．奇数番号の動画はスズメバチが 1 匹で，巣箱へ出入りするミツバチが少ない．偶数番号の動画はスズメバチが 2 匹で，巣箱へ出入りするミツバチが多い．全ての動画は 1fps で，表 6.3 の「スズメバチ数」は各動画の全フレームに映っているスズメバチの総数である．

表 6.3　精度評価動画

向き	ファイル名	解像度	フレーム数(枚)	スズメバチ数(匹)
正面	hornet1.mp4	1,280×720	54	42
	hornet2.mp4		391	580
斜め	hornet3.mp4	1,920×1,080	129	48
	hornet4.mp4		116	127
横	hornet5.mp4	1,280×720	16	11
	hornet6.mp4		121	135

- ImageAI (YOLOv3)

ImageAI V2.1.5 で事前に学習された YOLOv3 モデルから，データセット 1 を用いて転移学習を行った．学習済み YOLOv3 モデルは ImageAI のホームページからダウンロードした pretrained-yolov3.h5 を使用した．データセット 1 の 500 枚の画像をトレーニング用に 400 枚，テスト用に 100 枚に分けた．バッチサイズとエポック数は，ImageAI の公式ページに従って 4 と 200 に設定した．図 6.11 に学習回数と loss の関係を示す．エポック数を 200 としたが，学習開始から約 32 時間経過した 23steps 以降，2 日間学習を回し続けても TotalLoss が更新されなくなったので学習を途中で打ち切り，Loss が 3.602 と一番小さい 23step 目の学習データを用いた．その精度評価の結果を表 6.4 に示す．全体的な検出率は 88.9%と高いものの誤検出が多く，ミツバチや巣箱の木目等をスズメバチと検出し，図 6.12 のように何故か巣箱全体を誤検出することが多々あった．

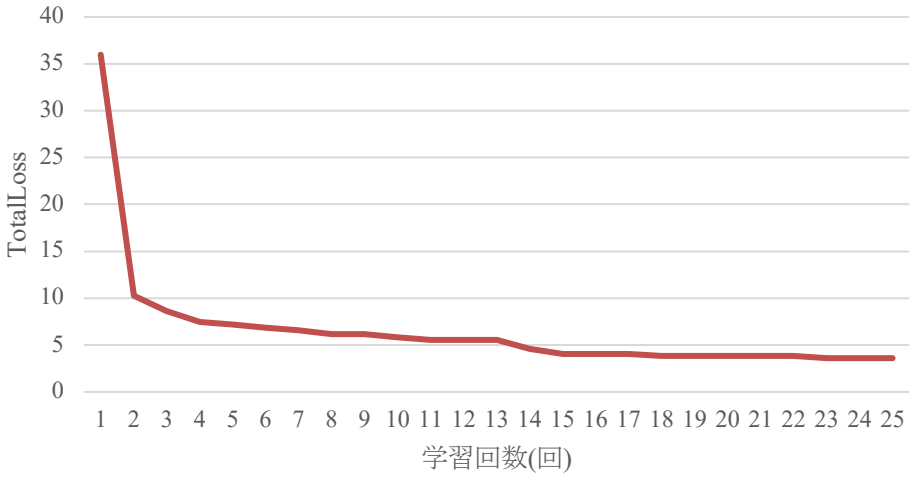


図 6.11　YOLOv3 の学習の様子

表 6.4 YOLOv3 with データセット 1

ファイル名	目視(匹)	AI 検出(匹)	AI 検出率(%)	AI 誤検出(回)
hornet1.mp4	42	40	95.2	0
hornet2.mp4	580	560	96.6	63
hornet3.mp4	48	39	91.2	49
hornet4.mp4	127	98	77.2	116
hornet5.mp4	11	8	72.8	0
hornet6.mp4	135	84	62.2	13
合計	943	829	88.9	241

図 6.12　YOLOv3 の誤検出(巣箱)

- Faster R-CNN (TensorFlow)

TensorFlow GPU 1.13.1 を用い，TensorFlow から提供されている Faster R-CNN の事前学習モデル faster_rcnn_inception_v2_coco [46]から転移学習を行った．データセット 1 の 500 枚の画像を，トレーニング用 400 枚とテスト用 100 枚に分け，バッチサイズは転移学習させるモデルで設定されている 1 を指定した．0.2 秒/step の速度で 92,133step 学習させた. 図 6.13 は Tensorboard[47]で学習回数と loss の関係をグラフ化したものである．このグラフの 92,133step 目の学習データで精度評価を行った結果を表 6.5 に示す．正面からの動画では検出率が 80%を超え誤検出も少ないが，斜めや横で検出率が低く，全体の平均は 61.0%であった．誤検出はカメラの近くを通ったミツバチをスズメバチと検出していた．その例を図 6.14 に示す.

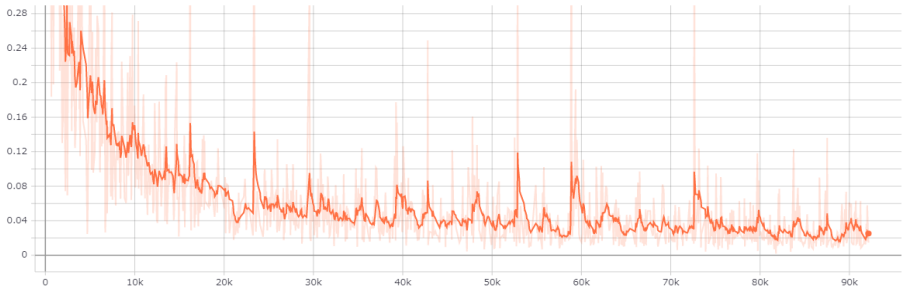


図 6.13　Faster-CCN の学習の様子

表 6.5　Faster R-CNN (データセット 1)

ファイル名	目視(匹)	AI 検出(匹)	AI 検出率(%)	AI 誤検出(回)
hornet1.mp4	42	33	78.6	0
hornet2.mp4	580	477	82.2	11
hornet3.mp4	48	6	12.5	0
hornet4.mp4	127	29	22.8	0
hornet5.mp4	11	0	0.0	0
hornet6.mp4	135	30	22.2	1
合計	943	575	61.0	12

図 6.14　Faster-CNN の誤検出(ミツバチ)

- SSDv1(TensorFlow Lite)

Faster R-CNN と同様に，TensorFlow から提供されている SSD 用の事前学習モデル ssd_mobilenet_v1_quantized_coco [46]から転移学習を行った．データセット 1 の 500 枚の画像をトレーニング用 400 枚とテスト用 100 枚に分けた．バッチサイズは，SSD は 24 が設定されていたが，そのままでは GPU がメモリ不足となってしまったため，エラーにならなくなった 6 を指定した．約 2 秒/step の速度で 100,000tep 学習させた際の学習回数と loss の関係を図 6.15 に示す．70,000step あたりで変化がなくなっているが，100,000step 目の学習データを TensorFlow Lite 用に変換して使用した．表 6.6 が精度評価の結果で，正面からの動画での検出率は約 48%と低く斜めと横は全く検出できなかった．

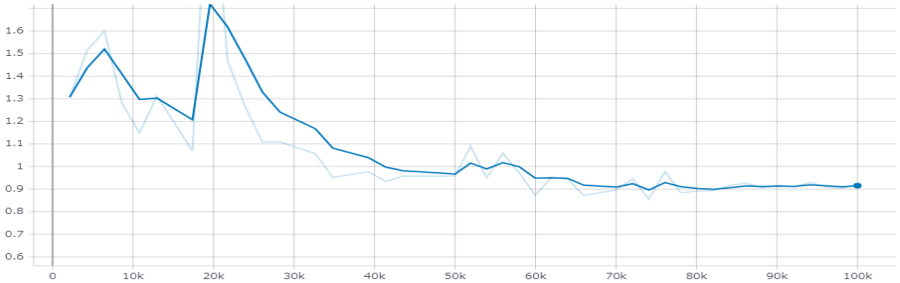


図 6.15　SSDv1 の学習の様子

表 6.6　SSDv1 (データセット 1)

ファイル名	目視(匹)	AI 検出(匹)	AI 検出率(%)	AI 誤検出(回)
hornet1.mp4	42	20	47.6	0
hornet2.mp4	580	282	48.6	0
hornet3.mp4	48	0	0.0	0
hornet4.mp4	127	0	0.0	0
hornet5.mp4	11	0	0.0	0
hornet6.mp4	135	0	0.0	0
合計	943	302	32.0	0

- SSDv2 (TensorFlow Lite)

SSDv1 と同様に, 事前学習モデル ssd_mobilenet_v2_quantized_coco[46]をデータセット 1 で 200,000step 転移学習させた．学習回数と loss の関係を図 6.16 に示す．Loss が 6 から学習が始まり 100,000step くらいから下がらなくなっているが，200,000step 目の学習データを TensorFlow Lite 用に変換して使用した．表 6.7 は精度評価結果で，正面の動画では hornnet1 と hornet2 を合わせた検出率は約 65%で SSDv1 よも高いが, 斜めと横はやはり検出しなかった.

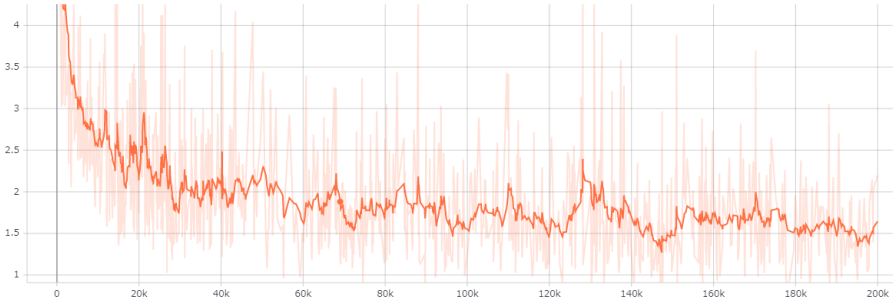


図 6.16　SSDv2 の学習の様子

表 6.7　SSDv2 (データセット 1)

ファイル名	目視(匹)	AI 検出(匹)	AI 検出率(%)	AI 誤検出(回)
hornet1.mp4	42	3	7.1	0
hornet2.mp4	580	399	68.8	0
hornet3.mp4	48	0	0.0	0
hornet4.mp4	127	0	0.0	0
hornet5.mp4	11	0	0.0	0
hornet6.mp4	135	0	0.0	0
合計	943	402	42.6	0

6.5　Raspberry Pi での速度評価

上記では，パソコン上で検出精度の評価を行ったが，実際には養蜂場に置いた Raspberry Pi による AI エッジコンピューティングシステムを構成することが目的である. そこで, 各環境を RP4 B 上に構築し, データセット 1 を用いて学習をした ImageAI, TensroFlow，TensorFlow Lite (SSDv2)モデルによる速度評価を行った．スズメバチが 2 匹映っている解像度 1,280×720 の 10 秒の動画 suzumebachi10s.mp4 に対する評価結果を表

6.8 に示す.

表 6.8　速度測定(fps)

	ImageAI	TensorFlow	TensorFlow Lite
suzumebachi10s.mp4	0.16	0.11	3.5

TensorFlow Lite では動画とカメラ共に 1 秒間に 3.5 フレームを処理することができたが，ImageAI は 1 枚のフレームに対して約 6 秒，TensorFlow は TensorFlow Lite の 30 倍の約 9 秒も要した．表 6.9 にモデル別のデータセット 1 の検出率(%)と誤検知(回)を示す．ImageAI と TensorFlow がサポートしている YOLOv3 や Faster R-CNN は，検出率が高いが，複数のカメラ映像を Raspberry Pi 上でリアルタイムにスズメバチを検出するのは不向きである.

表 6.9　モデル別データセット 1 の検出率(%)と誤検知(回)

ファイル名	YOLOv3		Faster R-CCN		SSDv1		SSDv2	
	検出率	誤検出	検出率	誤検出	検出率	誤検出	検出率	誤検出
hornet1.mp4	95.2	0	78.6	0	47.6	0	7.1	0
hornet2.mp4	96.6	63	82.2	11	48.6	0	68.8	0
hornet3.mp4	91.2	49	12.5	0	0.0	0	0.0	0
hornet4.mp4	77.2	116	22.8	0	0.0	0	0.0	0
hornet5.mp4	72.8	0	0.0	0	0.0	0	0.0	0
hornet6.mp4	62.2	13	22.2	1	0.0	0	0.0	0
合計	88.9	241	61.0	12	32.0	0	42.6	0

6.6　検出の精度向上

養蜂場に設置した Raspberry Pi 上でスズメバチをリアルタイムで検出するには，高速に実行できる TensorFlow Lite に対応した SSD モデルの利用が好ましい．また，全てのフレームでスズメバチを検出することよりも, 誤った警報が頻繁に利用者に届かないように誤検出をゼロにしたいという点においては，SSD は YOLOv3 や Faster R-CNN に対してある意味優れている言うこともできる．しかしながら，斜めと横の検出率がゼロというのでは, カメラの設置場所が変わった時にまったく検出できないという懸念が濃厚である．そこで，画像データを増やしたデータセット 2 とデータセット 3 を用いて，SSDv2 の精度向上を試みた．なお，評価にはこれまでと同じ 6 つの動画を用いた.

- データセット 2

データセット 2 の 1,000 枚の画像を，トレーニング用 800 枚とテスト用 200 枚に分け，バッチサイズを 6 と設定し，86,970tep 学習させた． 図 6.18 に学習回数と loss の関係を示す. 最後の 86,970step 目の学習データを TensorFlow Lite 用に変換して精度評価を行った結果を表 6.10 に示す. 正面からの動画ではデータセット 1 に対して検出率が 30%程度あがり，斜めからの動画では 3 回であったがスズメバチを検出した．しかし，ミツバチを 3 回誤検出し，横からの動画では相変わらず検出率がゼロであった.

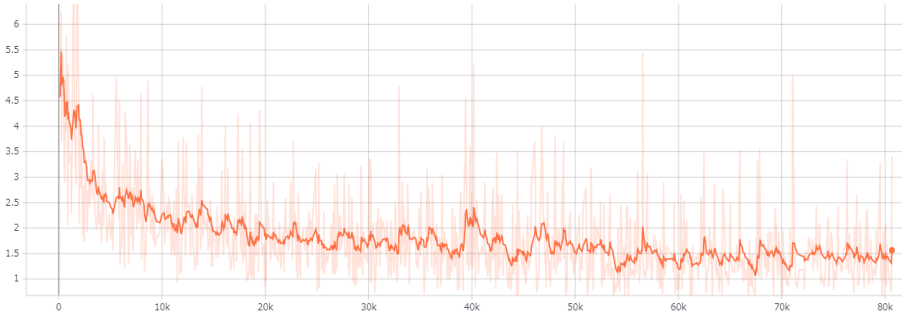


図 6.17　データセット 2 を用いた SSDv2 の学習の様子

表 6.10　SSDv2 (データセット 2)

ファイル名	目視(匹)	AI 検出(匹)	AI 検出率(%)	AI 誤検出(回)
hornet1.mp4	42	25	59.5	0
hornet2.mp4	580	498	85.9	3
hornet3.mp4	48	2	4.2	0
hornet4.mp4	127	1	0.8	0
hornet5.mp4	11	0	0.0	0
hornet6.mp4	135	0	0.0	0
合計	943	526	55.8	3

- データセット 3

データセット 3 の 1,400 枚の画像をトレーニング用 1,120 枚とテスト用 280 枚に分け，バッチサイズを 6 に設定して 89,621step 学習させた．図 6.18 に学習回数と loss の関係を示す．最後の 89,621step 目の学習データを TensorFlow Lite 用に変換して精度評価を行った結果を表 6.11 に示す．正面からの動画で検出率がデータセット 2 から 10%程度上がり，斜めと横からも検出率も 80%を超えた．しかし正面で誤検出が計 35 回も生じてしまった．hornet1 では巣箱の一部をスズメバチと誤検出していたが，推測の正しさは最大 62%で 50%以下が 27 回であった．hornet2 の誤検出はミツバチをスズメバチと誤検知していたが，推測の正しさは最大で 37%であった．つまりテストデータではスズメバチ検出の閾値を 62%よりも高く設定すれば誤検知をなくすことができる．先に述べたように検出率を上げるよりも誤検知をゼロにすることが重要なので，学習データを増やすことに加え，適切な閾値の設定方法を検討することで，さらなる精度の向上が狙える．

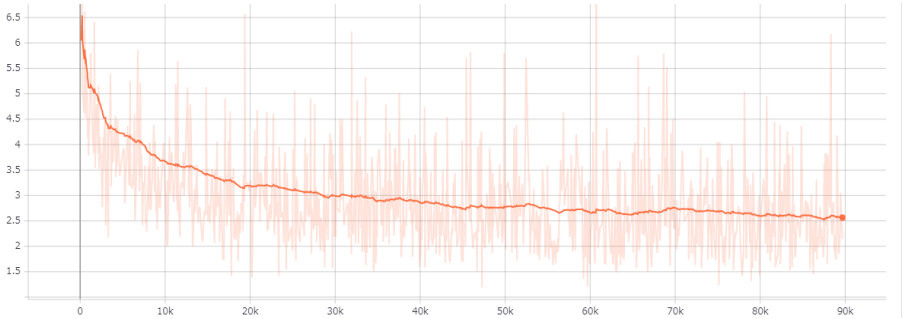


図 6.18　データセット 3 を用いた SSDv2 の学習の様子

表 6.11　SSDv2 (データセット 3)

ファイル名	目視(匹)	AI 検出(匹)	AI 検出率(%)	AI 誤検出(回)
hornet1.mp4	42	35	83.3	30
hornet2.mp4	580	559	96.4	5
hornet3.mp4	48	42	87.5	0
hornet4.mp4	127	110	86.6	0
hornet5.mp4	11	10	90.9	0
hornet6.mp4	135	120	88.9	0
合計	943	876	92.9	35

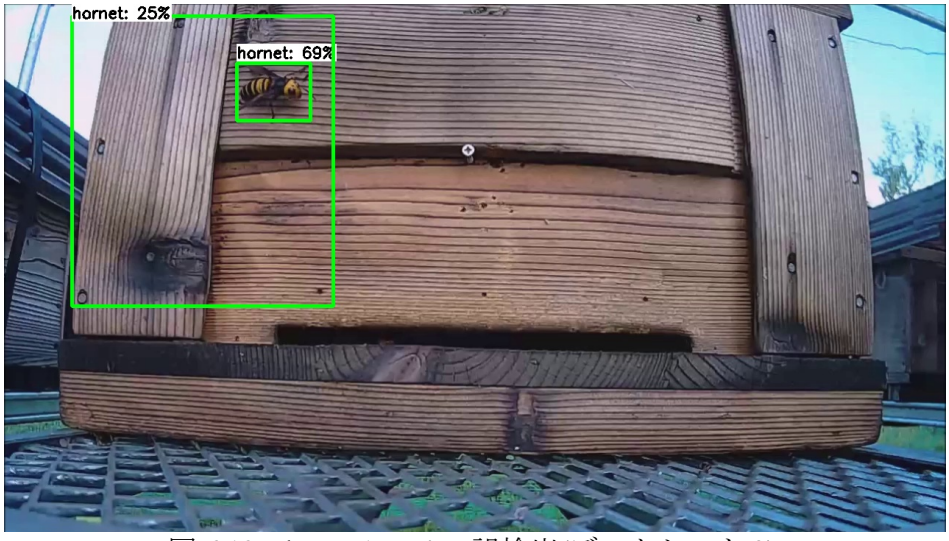


図 6.19　hornet1.mp4 の誤検出(データセット 3)

6.7　複数カメラによるスズメバチ検出

以下では Raspberry Pi に複数台のカメラを接続したときの，TensorFlow Lite の処理性能の評価を行う．その前に，6.5 節ではカメラ 1 台の動画に対して RP 4 B で 3.5fps の速度が出ることが示されたが，RP4 B より安価なモデルでの性能も評価し，対価格性能比でどのモデルを用いるべきかを調べておく．

まず予備実験として，RP4 B，RP3 B，RPZero W に TensorFlow Lite を実装し，10 秒のスズメバチ動画 suzumebachi10s.mp4 に対する処理性能を調べた．結果を Switch Sience での価格と共に表 6.12 に示す．RPZero W では 1 フレームあたり 16 (=1/0.059)秒以上の処理時間がかかっている．これは RPZero W は RP4 B のおよそ 1/6 の価格であるが，処理時間はおよそ 1/60 の性能である．また，RP3 B は RP4 B の 7 割の価格であるが，性能は 1/3 程度にとどまっている．従って，実用において，RP4 B が処理速度だけでなく，対価格性能比でも最も優れていることがわかる．

表 6.12　Raspberry Pi のと価格と速度の比較

モデル	RP 4 B	RP3 B	RPZero W
価格 (円)	7,700	5,775	1,320
処理性能 (fps)	3.5	1.3	0.059

次に，図 6.20~6.21 のように，RP4 B に WiFi ルータ経由で 1~10 台の ESP32 カメラを接続し，処理速度を調べる．スズメバチの写った巣箱の写真を ESP32 カメラで撮影し，800×600 ピクセルのデータを RTSP (Real Time Streaming Protocol)で RP4 B または RP3 B に転送し，SSDv2 モデルを用いた TensorFlow Lite で処理を行った．画像が FHD の 1920×1080 や HD の 1280×720 でなく 800×600 としたのは，Raspberry Pi の問題ではなく ESP32 のメモリサイズ制約からである．

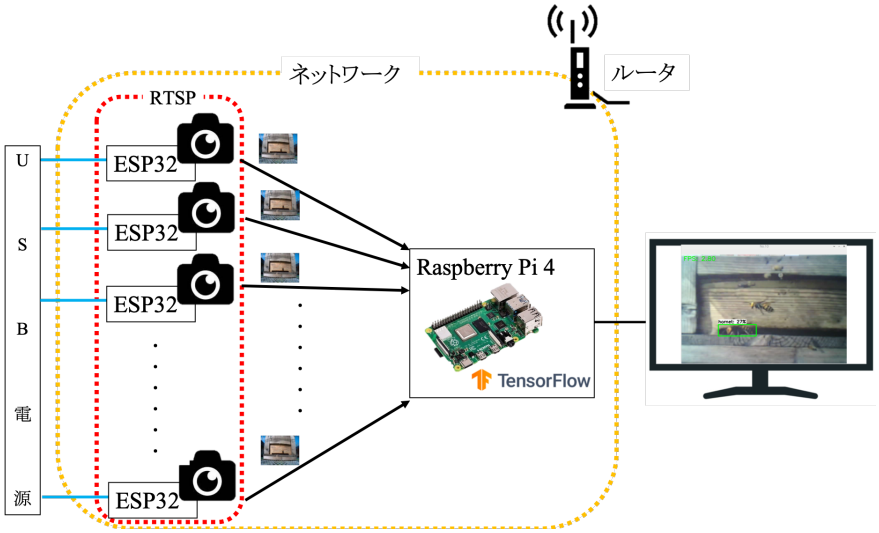


図 6.20　カメラと Raspbery Pi の接続関係

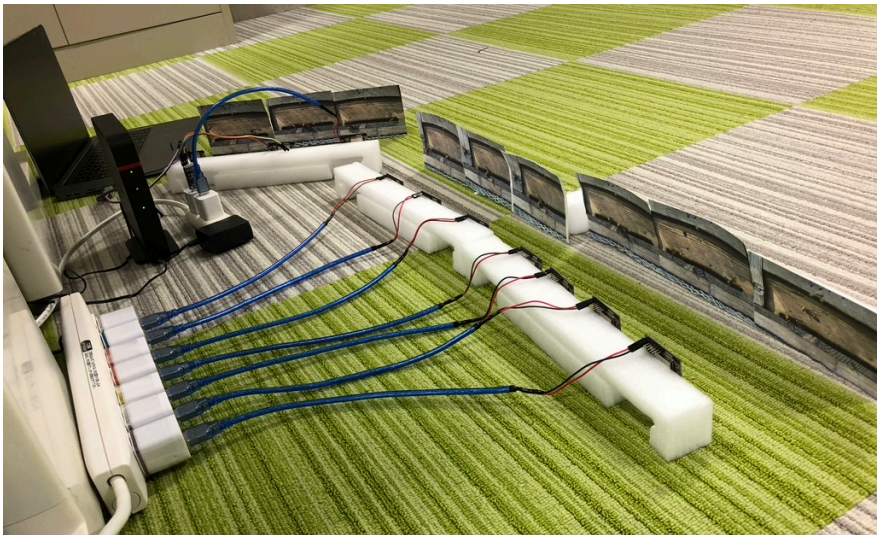


図 6.21　実験の様子

RP4 B，RP3 B，そして Coral USB Accelerator を挿した RP4 B に複数台のカメラを接続した場合の，カメラ台数，カメラ 1 台当たりの処理速度 fps/台，全体の処理速度 fps，そして CPU 使用率を表 6.13 と図 6.22 に示す．また図 6.23 はカメラを 6 台接続した際の RP4 B のモニタ画面である．

RP4 B と RP3 B は共に 4 コアなので，カメラの接続台数が 4 台までは，1 台当たりの

性能の低下はさほどない．そのため，台数が増えるにしたがって，トータルの fps は大きく向上する．RP3 B は 5 台目で CPU 使用率が 100%に達して処理速度が頭打ちとなり，8 台接続しようとするとフリーズしてしまった．それに対して RP4 B は 8 台目で CPU 使用率が 100%となり速度が頭打ちとなっているが，10 台接続しても止まることはなかった．またその時の 1 台当たりの処理速度は 1.05fps，つまり 1 秒に 1 枚の画像処理が行える．スズメバチは巣箱前に数分滞在することを考えれば，数十~数百枚の画像からスズメバチ検知が行えることになり，十分過ぎる性能と言える．従って今回はカメラが 10 台しか用意できなかったが，さらに増やせる可能性もある．また 10 台しか繋げなかったとしても，PP3 B よりも 3 割り高額なだけで，2 倍の性能が得られているため，RP4 B がより実用的である．

Coral USB Accelerator を使用した場合は，カメラが 1~2 台では RP4 B よりも高い性能が得られているが，それ以上ではほぼ互角である．11 台以上で両者の性能がどうなっていくかは不明ではあるが，Coral USB Accelerator の価格は 1 万円を超え，それを挿す Raspberry Pi が別途必要なことを考えれば，7,700 円の RP4 B を複数台用意したほうが有利であることは明白である．

表 6.13　RP3 B, RP4 B, Coral Accelerator の処理速度比較

接続台数	RP4 B			RP3 B			CoralAccelerator	
	fps/台	fps	CPU	fps/台	fps	CPU	fps/台	fps
1	2.51	2.51	18.6%	1.48	1.48	25.6%	8.33	8.33
2	2.43	4.46	40.2%	1.23	2.46	49.4%	3.56	7.12
3	2.31	6.93	40.9%	1.21	3.63	70.8%	2.21	6.63
4	2.14	8.56	71.0%	1.17	4.68	93.1%	1.76	7.04
5	1.89	9.45	78.5%	0.98	4.90	100%	1.61	8.05
6	1.50	9.00	82.0%	0.81	4.86	100%	1.56	9.36
7	1.47	10.29	91.6%	0.71	4.97	100%	1.38	9.66
8	1.29	10.32	100%	–	–	–	1.37	10.96
9	1.16	10.44	100%	–	–	–	1.15	10.35
10	1.05	10.05	100%	–	–	–	1.22	12.20

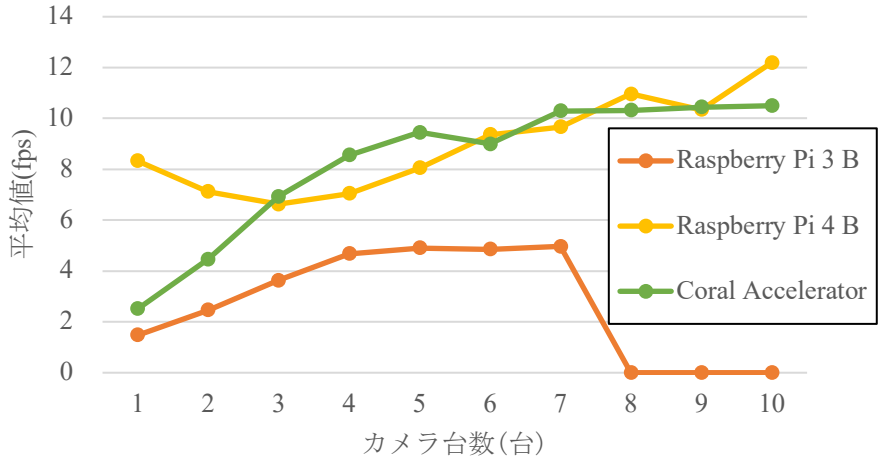


図 6.22　実験結果

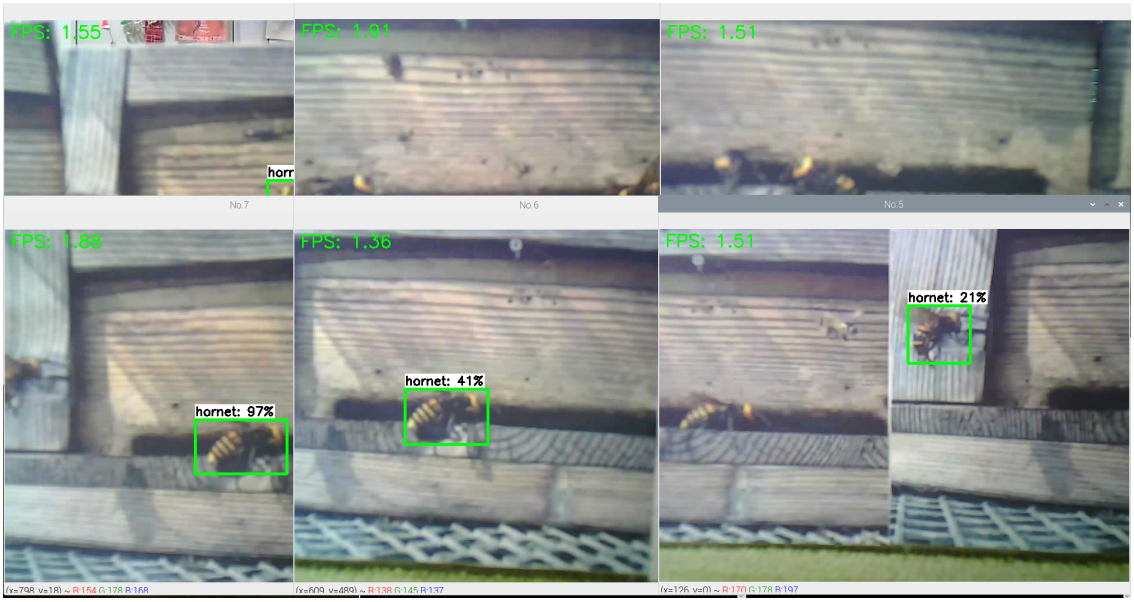


図 6.23　6 台のカメラを接続した際の RP4 B のモニタ画面

6.8　モバイルルータによる通信評価

養蜂場は草原や山間部などにあり，季節の花を求めて各地を転々とする移動養蜂もあるため，固定のブロードバンドネットワーク等は整っていないことが多い．そのためモバイルルータによる携帯電話回線の利用は必須である．しかし，モバイルルータは人が携帯して利用することが前提であるため，室内に固定して使用する Wi-Fi ルータに比べてローカルの通信距離は短い．そこで 6.7 章の実験環境の Wi-Fi ルータを表 6.14 の LTE モバイルルータ FREETEL ARIA 2 [48]に置き換え，ローカルの通信距離と通信速度が充分であるかどうかを屋外で調べた．

表 6.14 FREETEL ARIA 2 の諸元

外見	
型番	FTJ162A-ARIA2-W
通信速度	下り最大 150Mbps / 上り最大 50Mbps
同時接続台数	最大 10 台
無線 LAN	IEEE 802.11 b/g/n (2.4GHz)
バッテリー	容量 2,300mAh / 連続通信時間：17 時間

ESP32 は 3D プリンタで自作したケースに，Raspberry Pi とルータは 防水ボックスに入れて，ESP32 と防水ボックスの距離を 1m ずつ離してフレームレートを測定した．図

6.24 に実験の様子，測定結果を図 6.25 に示す．距離が 8m までは安定して 3fps 以上のレートで通信ができていたが，9m 以上では通信が不安定となった．養蜂箱はある程度まとめて設置するので，その中心にルータを設置すれば半径 8m は十分な距離であると言える．

しかしながら電源の確保が課題として残っており，大容量モバイルバッテリー，カーバッテリー，そして太陽光発電の利用等を検討している．そして，WeMos D1 R32 ボードの子基板として，発電量と消費量が遠隔モニタ可能な図 6.26 のソーラチャージコントローラも開発中である．

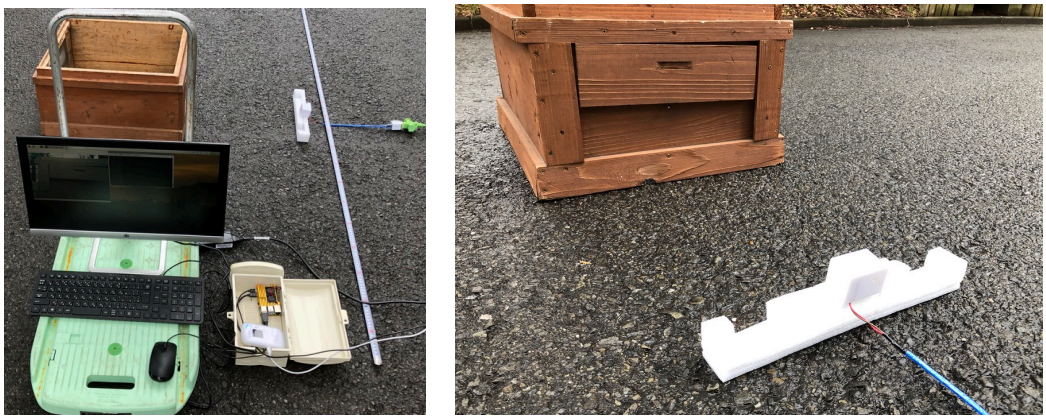


図 6.24　1m 離したときの実験の様子

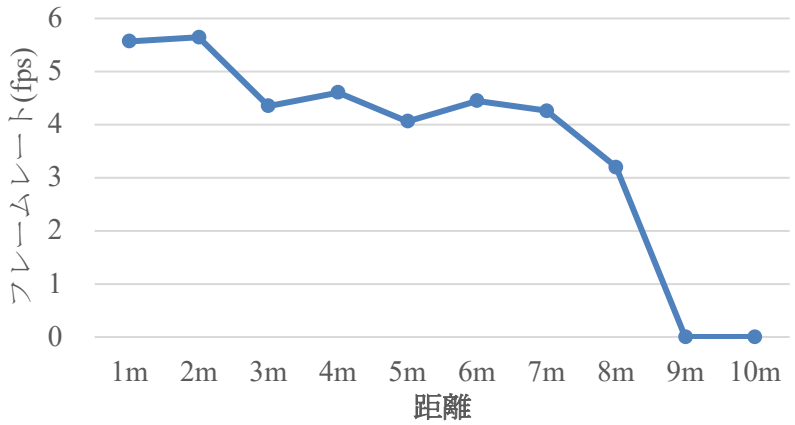


図 6.25 測定距離とフレームレート

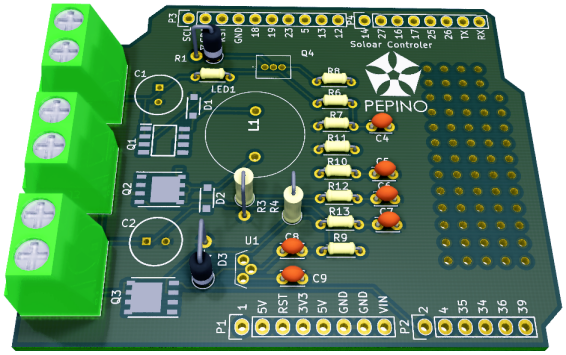


図 6.26　開発中のソーラチャージコントローラのイメージ図

7　ミツバチの個体数計測

7.1　概要

3.2 節で紹介したミツバチの巣門前の個体数計測に関する先行研究[20][21]では，ミツバチの検出能力が低いため誤差が 2 倍以上という問題があった．6 章では襲来する数匹のスズメバチの検出を行っていたが，この機械学習モデルをミツバチの検出に適用することで，数十匹の個体のカウントを行いその精度を評価する．

7.2　学習用データセットの生成

個体数計測に使用するデータセットは，赤坂みつばちあの巣箱の巣門を上から，巣箱内の撮影に用いた赤外線カメラと同じもので撮影した動画を用いた．RP3 B に赤外線カメラとデータ保存用外付け HDD を RP3 B に接続し，コマンドツール Raspivid で Full HD で 30fps の動画を取得した．Raspivid は H.264 コーデックのみ対応しているため，保存した動画データを後日, MP4Box で MP4 フォーマットに変換した. 撮影期間は 2019 年 9 月 17 日 18 時~19 日 6 時で，その動画から次の 2 つのデータセットを作成した．

- データセット 4

データセット 4 は 1,000 枚のラベル付き画像である．図 7.1 にデータセット 4 の作成手順を示す．1,920×1,080 の画像を 320×320 のエリアに分割して，ミツバチが映っている画像を 1,000 枚選別した後にアノテーションを行なった． 図 7.2 にデータセット 4 の一部を示す．

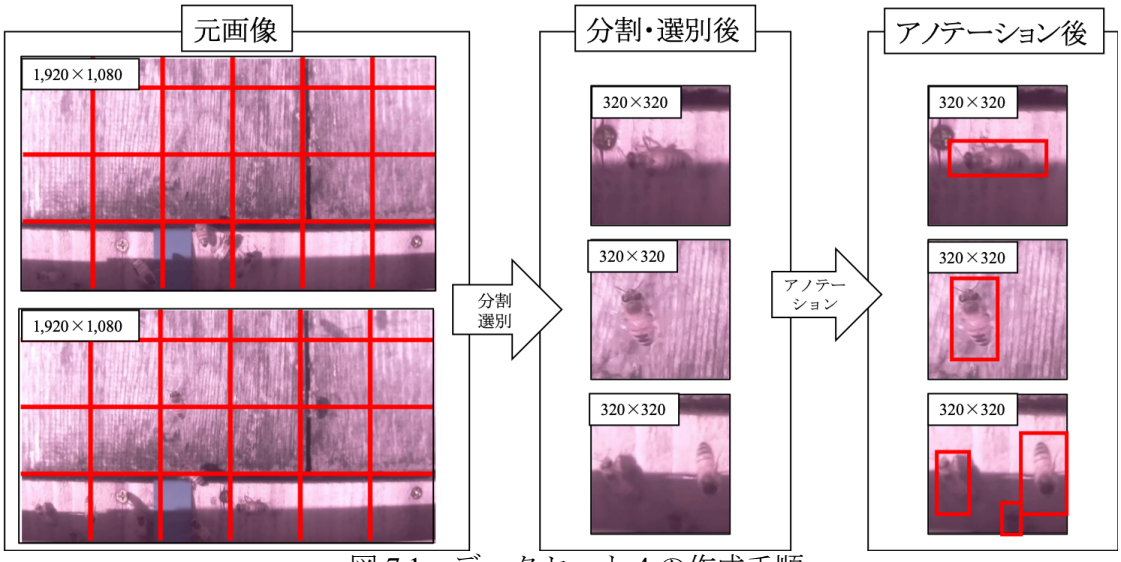


図 7.1　データセット 4 の作成手順

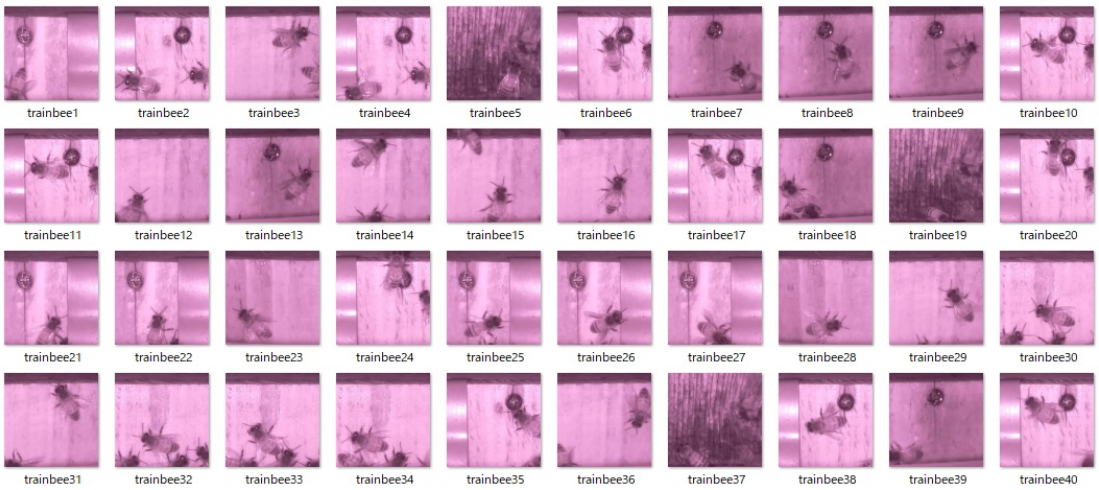


図 7.2　データセット 4 の画像例

- データセット 5

データセット 5 は 100 枚のラベル付き画像である．図 7.3 にデータセット 5 の作成手順を示す．1,920×1,080 の画像を切取りせずに 480×270 に縮小し，アノテーションを行なった．図 7.4 にデータセット 5 の一部を示す．

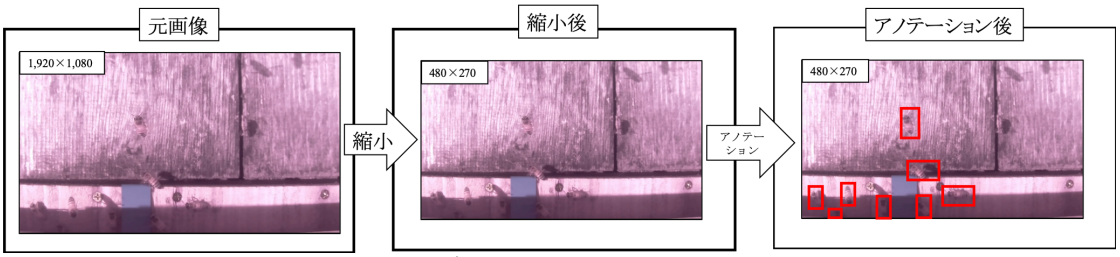


図 7.3　データセット 5 の作成手順

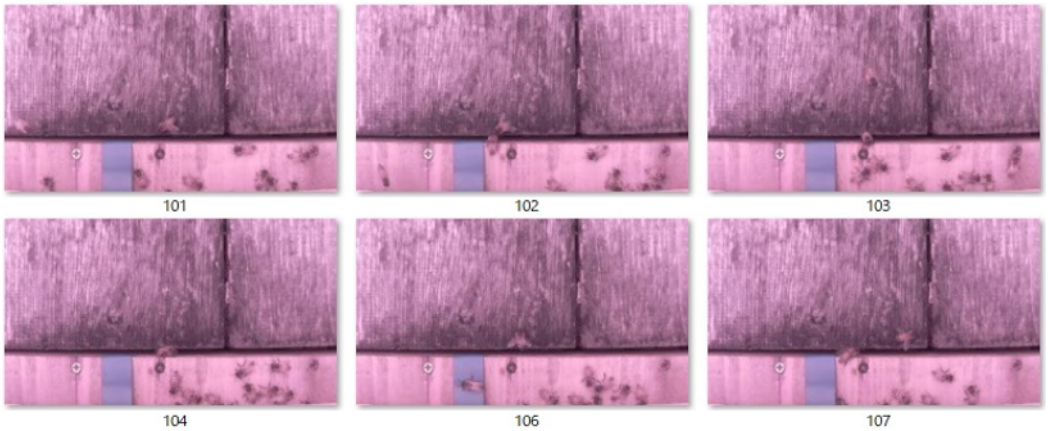


図 7.4　データセット 5 の画像例

7.3　機械学習によるミツバチの個体数計測

データセット 4 と 5 を用いて，6 章でも使用した Faster R-CNN と SSDv2 の学習済みモデルから転移学習を行った．精度評価にはデータセットに使用していない動画から，ミツバチが 1 フレーム中に 15 匹未満の bee1.mp4 と，15 匹以上映っている bee2.mp4 を

使用した．いずれも解像度 1,920×1,080，フレームレート 1fps，100 秒の動画である．

● データセット 4 による学習

データセット 4 の 1,000 枚の画像をトレーニング用 800 枚とテスト用 200 枚に分け，Faster R-CNN モデルはバッチサイズ 1 で 200,000 回，SSDv2 モデルはバッチサイズ 6 で 38,526 回学習させた．その結果はいずれも図 7.5 のように，ミツバチ個体より大きなボックスで検出され，未検出も多い結果となった．

図 7.5　データセット 4 の結果

● データセット 5 による学習

データセット 5 の 100 枚の画像をトレーニング用 80 枚とテスト用 20 枚に分け，Faster R-CNN モデルでバッチサイズを 1 で 200,000 回，SSDv2 モデルはバッチサイズ 6 で 50,306 回学習させた．学習した各モデルで動画 bee1.mp4 と bee2.mp4 に対するミツバチの検出を行わせたところ，図 7.6~7.9 のように良好な結果が得られた．

図 7.6　Faster R-CNN によるミツバチの検出 (bee1.mp4)

図 7.7　SSDv2 によるミツバチの検出 (bee1.mp4)

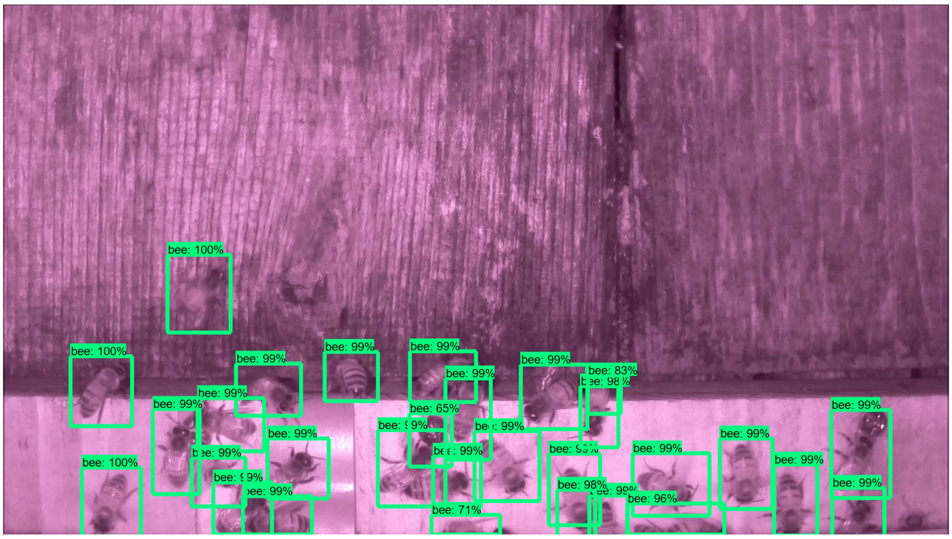


図 7.8　Faster R-CNN によるミツバチの検出(bee2.mp4)

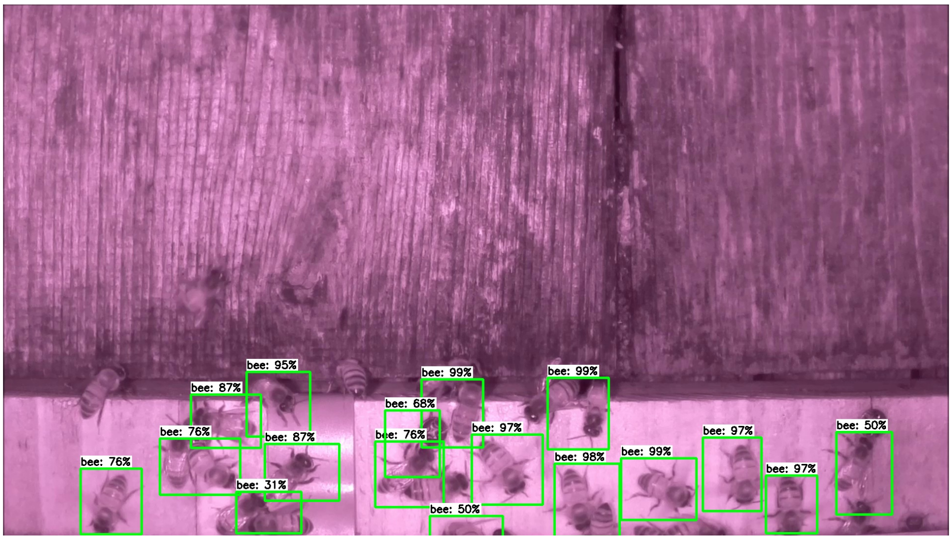


図 7.9　SSDv2 によるミツバチの検出 (bee2.mp4)

そこで次に，ミツバチの数をカウントさせ，人の目視によるカウント数と比較した．動画 bee1.mp4 に対する Faster R-CNN と SSDv2 の結果をそれぞれ図 7.10 と図 7.11 に示す．機械学習によるカウントと目視のグラフはほぼ一致しており，異なっていても差は 2 匹が各 1 回，それ以外はわずか 1 匹である．

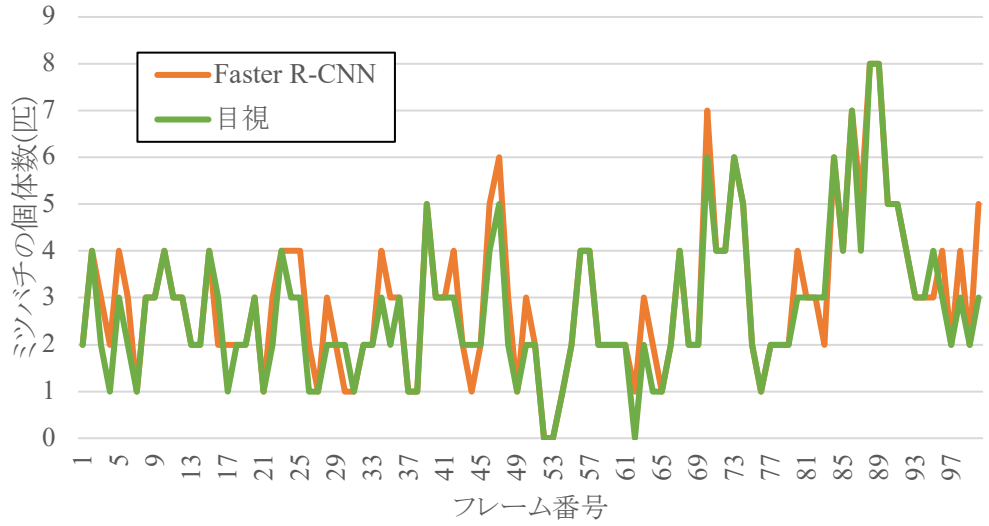


図 7.10　Faster R-CNN によるミツバチのカウント(bee1.mp4)

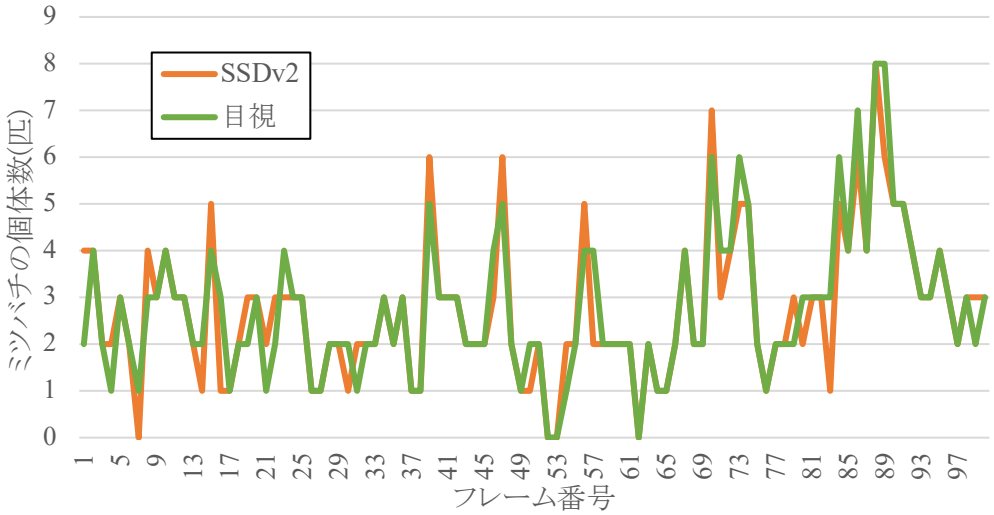


図 7.11　SDDv2 によるミツバチのカウント(bee1.mp4)

次に bee2.mp4 に対するカウント数を図 7.12~7.13 に示す．Faster R-CNN モデルは正解が 21%，誤差 5 匹未満が 63%，誤差 5 匹以上が 6%で，SSDv2 モデルは正解が 10%，誤差 5 匹未満が 51%，誤差 5 匹以上が 39%となった．また，図 7.12 から Faster R-CNN は全体的に多くカウントしているが，図 7.13 から SSDv2 は全体的に少なくカウントしていることがわかる．図 7.8 の Faster R-CNN の bee2.mp4 に対するミツバチ検出の結果から，背景の木目を誤検出したり，1 匹のミツバチをダブルカウントしていることがカウント数の多い原因とわかる．また SSDv2 のカウントが少ないのは図 7.9 から，背景の木目と同化しているミツバチが検出できないことが原因である．

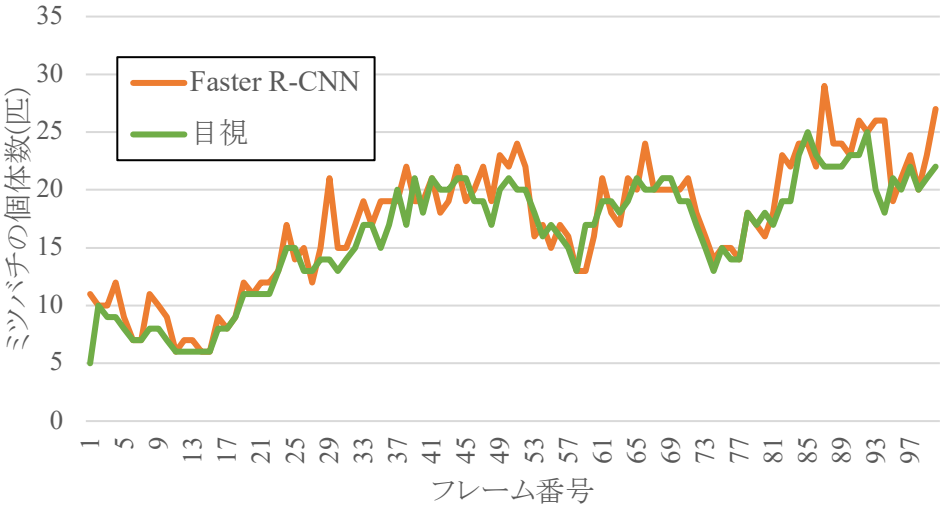


図 7.12　Faster R-CNN によるミツバチのカウント(bee2.mp4)

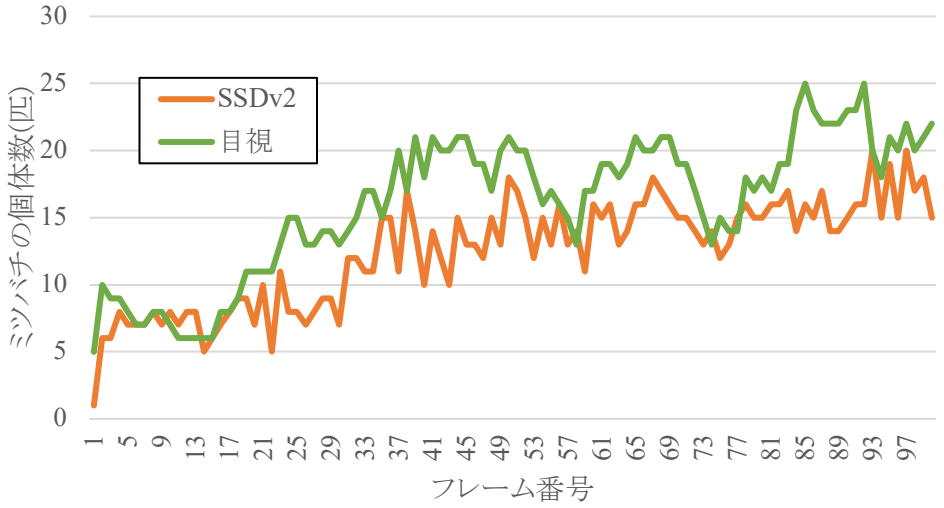


図 7.13　SDDv2 によるミツバチのカウント(bee2.mp4)

表 7.1 は，2 つの動画それぞれの 100 フレームに映っているミツバチの総数を，人の目視によるカウントと機械学習モデルによる検出数，そしてその割合についてまとめたものである．Faster R-CNN は 10%未満の誤差で，SSDv2 は個体数が多い場合の誤差は 2 割を超えてしまったが，従来研究の半分以下の誤差であり，極めて大きな改善が見られる．また，データセット 5 はデータセット 4 の 1/10 の画像枚数にも関わらず，高い精度で検出した．スズメバチの検出も同様だが，検出対象を切り取った画像よりも全体の画像で学習させることが効果的である．それは全体を映すことでミツバチでない部分についても学習され，全体画像によって個体のサイズまで含めて学習が進むためと考えられる．特に養蜂ではカメラをある程度決まった距離や角度に固定して使用するので，全体の画像での学習は有効である．

まだ初期の実験の段階であり，今後様々な改善により精度の向上が可能である．例えば，個体一つ一つを識別する必要はあまりなく，全体で何匹という情報が重要なため，

誤差も見越したうえで補正をかけるということができる．また，巣門前の個体数ではなく，採蜜のために出入りするミツバチの数だけをカウントすることで，活動状態を調べたり，巣箱を開けて巣枠に密集している蜂の状態を全体で観察して群の個体数を把握したりといったことが考えられる．このように，カメラ動画と機械学習を用いた研究の養蜂への応用範囲は広い．

表 7.1　各アルゴリズムの誤差

動画ファイル	目視	Faster R-CNN			SSDv2		
	総数(匹)	総数(匹)	誤差(匹)	誤差(%)	総数(匹)	誤差(匹)	誤差(%)
bee1.mp4	275	297	+22	+8.0	272	-3	-1.1
bee2.mp4	1,610	1,716	+106	+6.6	1,249	-361	-22.4

8 むすび

本研究ではIoT技術をこれまで経験とカンに頼ってきた養蜂に導入し，巣箱の内外の状態を遠隔でモニタするシステムの開発とその評価を行った．外気および巣箱内の温湿度，重量，二酸化炭素濃度を計測し，スマートフォン等のモバイルデバイスでその変化をモニタするセンサシステム，そして早期対応が必須であるスズメバチの襲来を機械学習モデルによって検出するカメラシステムを IoT 用マイコンボード ESP32 および Raspberry Pi を用いて実装した．

温湿度センサ，重量センサ，二酸化炭素センサを設置した養蜂箱では，ダニの発生や分蜂等が起こらなかったため異常検知はできなかった．しかし，遠隔モニタで巣箱の温湿度や二酸化炭素濃度は一定に保たれ正常であること，重量変化はミツバチの群勢や採蜜時期の判断材料としての有用性が示された．

機械学習を用いたスズメバチの検出では Web の写真やビデオ撮影した動画から切り出した画像データを用いて，ImageAI, TensorFlow, TensorFlow Lite 環境で比較評価した．TensorFlow Lite の検出率は非常に低かったが，処理速度は ImageAI の 20 倍，TensorFlow の 30 倍以上と非常に高速であった．そこで，新たな学習データを加えることで，83~96%の高検出率が得られた．誤検出も多少生じたが，検出の閾値を高くすることでゼロにできる可能性を示した．

また，最大 10 台のカメラを接続した場合の処理性能について，Raspberry Pi の各モデルやエッジ AI アクセラレータを比較し，対価格性能比も含めて Raspberry Pi 4 の高い有意性が示された．またモバイルルータを用いた通信距離の評価では，半径 8m と十分な性能が得られることも実験で示した．

さらに，スズメバチの検出の手法をミツバチの個体数のカウントに応用し，誤差 7~22%という結果が得られた．精度向上に向けて多くの改善の余地が残されているが，先行研究の誤差 2 倍以上と比較して極めて高い精度である．この結果は IoT や AI を応用した養蜂の研究が，いかに初期の段階であるかを物語っている．

機械学習によるミツバチの映像解析は，巣枠に密集しているミツバチの状態を全体で観察して群の個体数を把握したり，採蜜のために出入りするミツバチだけをカウントし，また働きバチ毎に決まっている役目を検出・分類することで活動状態を調べるなど様々な応用が考えられる．本研究はその大きな可能性を示すことができた．

参考文献

[1] 安岡澄人: スマート農業の推進, 日本ロボット学会誌 Vol.35 No.5, pp362-365, 農林水産省, 2017年.

[2] A. Satoh: A Hydroponic Planter System to Enable an Urban Agriculture Service Industry, Proc. IEEE 7th Global Conference on Consumer Electronics (GCCE 2018), OS-ICE(1)-1, Oct. 2018.

[3] 小池誠: ディープラーニングを用いたキュウリ選果機の開発, 2017年9月.
https://aitc.jp/events/20170919-Seika/20170919_招待講演_ディープラーニングを用いたキュウリ選果機の開発_AITC.pdf（参照 2019-1-25）

[4] The Raspberry Pi Foundation: Teach, Learn and Make with Raspberry Pi.
https://www.raspberrypi.org/（参照 2019-1-25）

[5] Arduino – Home.
https://www.arduino.cc（参照 2019-1-25）

[6] 国連広報センター:持続可能な開発目標(SDGs).
https://www.unic.or.jp/activities/economic_social_development/sustainable_development/2030agenda/（参照 2019-1-25）

[7] 特定非営利活動法人銀座ミツバチプロジェクト:銀座ミツバチプロジェクト.
http://www.gin-pachi.jp/（参照 2019-1-25）

[8] 鹿島建設株式会社:鹿島ニホンミツバチプロジェクト.
https://www.kajima.co.jp/news/digest/jul_2009/kensaku/index-j.htm（参照 2019-1-25）

[9] 株式会社TBSテレビ:みつばちプロジェクト.
https://www.tbs.co.jp/csr/mitsubacheer/mitsubacheer.html（参照 2019-1-25）

[10] 株式会社Beetopiaはらじゅく:じんぐうまえハニープロジェクト.
https://beetopia.tokyo/（参照 2019-1-25）

[11] 特定非営利活動法人サッポロ・ミツバチ・プロジェクト:サッポロ・ミツバチ・プロジェクト.
http://sappachi.com/（参照 2019-1-25）

[12] 仙台ミツバチプロジェクト: 仙台ミツバチプロジェクト.
https://sen-pachi.jp/（参照 2019-1-25）

[13] 山田順之:ニホンミツバチプロジェクト ～都市域における生物多様性への取組み～, アーバンアドバンスNo.52, pp40-46, 2010年6月.

[14] 農林水産省生産局畜産部:養蜂をめぐる情勢(2019年11月)
https://www.maff.go.jp/j/chikusan/kikaku/lin/sonota/attach/pdf/hachimeguji.pdf
（参照 2019-1-25）

[15] 渡辺寛, 渡辺孝:近代養蜂 改訂第6版, 日本養蜂振興会出版, 2019年

[16] OSBeehives: OSBeehives | BuzzBox Hive Health Monitor & Beekeeping App.

https://www.osbeehives.com/（参照 2019-1-25）
[17] 3Bee: Technology for beekeepers | 3Bee.
https://www.3bee.it/en/（参照 2019-1-25）
[18] アドダイス株式会社: IoTとAIを利用した養蜂業向けテクノロジー個性豊かで生産者の見える「四季の蜂蜜」の紹介.
https://bee-sensing.com/（参照 2019-1-25）
[19] ミツバチサミット実行委員会: シンポジウム8(養蜂)「養蜂の現状と未来」.
https://bee-summit.jp/timetable/event/シンポジウム08（参照 2019-1-25）
[20] V. Kulyukin and S. Reka: A Computer vision algorithm for omnidirectional bee counting at langstroth beehive entrances, Proc. Intl. Conf. on Image Processing, Computer Vision, and Pattern Recognition (IPCV'16), pp. 229-235, Jul. 2016.
[21] V. Kulyukin and S. Mukherjee: On Video Analysis of Omnidirectional Bee Traffic: Counting Bee Motions with Motion Detection and Image Classification, Applied Sciences, vol. 9, no. 18, 3743, Sep. 2019.
[22] K. Shimasaki, M. Jiang, T. Takaki, I. Ishii, and K. Yamamoto: HFR-Video-Based Honeybee Activity Sensing Using Pixel-Level Short-Time Fourier Transform, Proc. IEEE Sensors 2018, pp.160-163, Oct. 2018.
[23] 広島大学大学院工学研究科システムサイバネティクス専攻ロボティクス研究室: 高フレームレートビデオベース蜜蜂アクティビティセンシング,
http://www.robotics.hiroshima-u.ac.jp/vision_sensing/STFT-j.php（参照 2019-1-25）
[24] 島田隆介: 蜜蜂の活動状態認識のための動画解析の研究, 平成30年度電気通信大学卒業論文, 2018年.
[25] ATMEL: 8-bit PIC and AVR MCUs.
https://www.microchip.com/design-centers/8-bit（参照 2019-1-25）
[26] Espressif Systems: Espressif Systems - Wi-Fi and Bluetooth chipsets and solutions.
https://www.espressif.com/（参照 2019-1-25）
[27] LilyGo: lilygo.
http://www.lilygo.cn/（参照 2019-1-25）
[28] 深圳市安信可科技有限公司: AI Thinker.
https://www.ai-thinker.com/home（参照 2019-1-25）
[29] Google: Coral.ai.
https://coral.ai（参照 2019-1-25）
[30] Intel Corporation: Intel Neural Compute Stick 2 | Intel Software.
https://software.intel.com/en-us/neural-compute-stick（参照 2019-1-25）

[31] R. Girshick, J. Donahue, T. Darrell, and J. Malik: Rich feature hierarchies for accurate object detection and semantic segmentation, Proc. the IEEE conference on computer vision and pattern recognition (CVPR2014), pp.580-587, Jun. 2014.

[32] R. Girshick: Fast R-CNN, Proc. the IEEE International Conference on Computer Vision (ICCV 2015), pp. 1440-1448, Sep. 2015.

[33] S. Ren, K. He, R Girshick, and J Sun: Faster R-CNN: Towards real-time object detection with region proposal networks, Advances in Neural Information Processing Systems (NIPS 2015). Jun. 2015.

[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi: You only look once: Unified, real-time object detection, Proc IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2106), pp. 789-798, Jun. 2016.

[35] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg: SSD: Single Shot MultiBox Detector, Proc. European Conference on Computer Vison (ECCV 2016), pp.21-37, Sep. 2016.

[36] M. Olafenwa: Official English Documentation for ImageAI! — ImageAI 2.1.5 https://imageai.readthedocs.io/en/latest/（参照 2019-1-25）

[37] TensorFlow: TensorFlow. https://www.tensorflow.org/（参照 2019-1-25）

[38] TensorFlow:TensorFlow Lite. https://www.tensorflow.org/lite?hl=ja（参照 2019-1-25）

[39] 株式会社オーム電機:体重計. https://www.ohm-electric.co.jp/product/c1605/（参照 2019-1-25）

[40] 気象庁: 過去の気象データ検索. https://www.data.jma.go.jp/obd/stats/etrn/index.php?prec_no=44&block_no=47662&year=&month=&day=&view=（参照 2019-1-25）

[41] E. Southwick and R. Moritz, “Social control of air ventilation in colonies of honey bees, apis mellifera,” Journal of Insect Physiology, vol. 33, no. 9, pp. 623 – 626, 1987.

[42] 気象庁: 二酸化炭素濃度の観測結果. https://ds.data.jma.go.jp/ghg/kanshi/obs/co2_monthave_ryo.html（参照 2019-1-25）

[43] 株式会社坊ノ内:bee slow. http://beeslow.com（参照 2019-1-25）

[44] SpotCam:家庭用無線カメラ. https://www.myspotcam.com/jp（参照 2019-1-25）

[45] darrenl: labelImg. https://github.com/tzutalin/labelImg（参照 2019-1-25）

[46] TensoeFlow: Tensorflow detection model zoo – GitHub.
https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md（参照 2019-1-25）
[47] TensorFlow: TensorBoard | TensorFlow.
https://www.tensorflow.org/tensorboard（参照 2019-1-25）
[48] FREETEL: ARIA 2.
https://www.freetel.jp/product/wifi/aria2/（参照 2019-1-25）

謝辞

本研究を進めるにあたり，指導教官の佐藤証教授から丁寧かつ熱心なご指導を賜り，心より感謝いたします．また，研究にご協力いただき，ミツバチの凄さや養蜂の楽しさを教えていただいた「赤坂みつばちあ」，「BEETOPIA はらじゅく」，「株式会社坊ノ内」の皆様にも心より感謝致します．